LAUREA

# Regressiotestauksen kehittäminen yrityksessä X

. . . . . .

Lax, Samu

2009 Laurea Leppävaara

**Laurea-ammattikorkeakoulu**
Laurea Leppävaara

# Regressiotestauksen kehittäminen yrityksessä X

Samu Lax
Tietojenkäsittelyn koulutusohjelma
Opinnäytetyö
Marraskuu 2009

**Laurea-ammattikorkeakoulu** **Tiivistelmä**
Laurea Leppävaara
Tietojenkäsittelyn koulutusohjelma
Yritysten tietoverkot

Samu Lax

**Regressiotestauksen kehittäminen yrityksessä X**

Vuosi 2009 Sivumäärä 40

Opinnäytetyössä tutkitaan yrityksen X regressiotestausprosessia ja tehdään havaintoja CA Verify Automated Regression Testing for CICS -testaustyökalun soveltuvuudesta yritykseen.

CA Verify Automated Regression Testing for CICS -testaustyökalu hankittiin yrityksen X koekäyttöön, koska regressiotestauksessa oli todettu puutteita. Työkalu on automaattinen testaustyökalu pääasiassa regressiotestauksen helpottamiseen. Regressiotestaus on aikaisemmin testatun sovelluksen testaamista uusilla versioilla sovelluksesta. Tavoite on varmistaa, että toiminnot, jotka toimivat edellisissä versioissa toimivat edelleen odotetusti. Näin sen voidaan katsoa olevan laadun varmistusta pitkällä tähtäimellä. Regressiotestaus on tutkitusti yleisimpiä testauksen kohteita, johon testaustyökalu hankitaan. Testaustyökalua hankittaessa tulee ottaa huomioon useita seikkoja. Ilmeisin syy testausautomaatioon sijoittamisessa on mahdollisesti saavutettavat kustannussäästöt. Työkalua ei kuitenkaan kannata hankkia kaikentyyppisiin testeihin tai järjestelmiin. Onnistuneissa tapauksissa työkalusta on saavutettu huomattavaa hyötyä, kun taas joissain yrityksissä hankittu työkalu on jäänyt käyttämättä vähäisten hyötyjen tai huonon käytettävyyden takia.

Teoriaosuudessa käsitellään testausta yleisesti osana ohjelmistoprosessia ja keskitytään erityisesti regressiotestaukseen. Lisäksi tarkastellaan testausautomaatiota sekä sen mahdollisia etuja ja haittoja.

Tutkimusosa esittelee yrityksen regressiotestauksessa havaittuja seikkoja ja CA Verify Automated Regression Testing for CICS -työkalua, työkalun koekäytön kulkua ja sen lopputuloksia.

Tutkimustulosten perusteella voitiin päätellä, että yrityksen regressiotestausprosessissa on parannettavaa. CA Verify Automated Regression Testing for CICS -työkalun koekäyttö osoitti, että kyseinen työkalu saattaisi olla ratkaisu regressiotestauksessa koettuun ajan puutteeseen. Lisäksi sen käyttöä voitaisiin laajentaa useisiin eri testausmahdollisuuksiin.

Asiasanat: testaus, testausautomaatio, regressiotestaus

**Laurea University of Applied Sciences**
Laurea Leppävaara
Information Technology programme
Information Systems and Data Communication

**Abstract**

Samu Lax

**Developing regression testing in company X**

Year 2009 Pages 40

The purpose of this thesis is to study the regression testing process of company X and make observations about the suitability of the CA Verify Automated Regression Testing for CICS testing tool for the company.

The CA Verify Automated Regression Testing for CICS testing tool was acquired for trial because there were known shortages in the regression testing process of company X. The tool is an automated testing tool which is mainly used to make regression testing easier. Regression testing means testing a previously tested application with new versions of the application. The purpose is to ensure that the functions which worked in the previous versions still work as expected. Thus it can be considered as quality assurance in the long run. Regression testing is one of the most common areas where test automation is issued. When acquiring a testing tool, several issues are to be taken into account. The most obvious reason to acquire a testing tool is possible cuts in testing expenses. However, not every type of test or system is suitable for test automation though. In successful cases, the benefits of the acquired tool are significant, whereas in other cases the tool has remained unused because of lack of benefits or usability.

The theory section examines testing generally as a part of the program development process and focuses especially on regression testing. Test automation and its possible advantages and disadvantages are considered in this section.

The issues observed while studying the regression testing process of the company are presented in the research section. Also the course and the results of the trial of the CA Verify Automated Regression Testing for CICS testing tool are included here.

The results of the study show that there are some shortages in company X's regression testing process. The trial of the CA Verify Automated Regression Testing for CICS pointed out that the tool could be one solution to solve the lack of time in regression testing in the company. In addition, its use could also be extended to cover other types of testing.

Keywords: system testing, test automation, regression testing

Sisällys

# 1 Johdanto

Tässä opinnäytetyössä tutkitaan yrityksen X regressiotestausta ja miten uuden testaustyökalun, CA Verify Automated Regression Testing for CICS:in, koekäyttö vaikuttaa regressiotestausprosessiin. Tarkoituksena on tehdä havaintoja nykyisestä prosessista, löytää mahdollisia kehitystarpeita sekä toimia yllä mainitun testaustyökalun koekäyttäjänä. Tämän perusteella arvioidaan, onko työkalun hankkiminen yritykselle kannattavaa. Regressiotestaus on järjestelmätestaukseen liittyvä käsite, jolla tarkoitetaan sovelluksen uudelleentestaamista muutoksen jälkeen. Regressiotestauksella varmistetaan ohjelman ajantasaisuus ja oikea toimivuus.

Yritys X on IT-alan yritys, joka tarjoaa tietotekniikkapalveluja tietoliikenne- ja mediasektorille. Palveluvalikoima kattaa tutkimus- ja kehityspalvelut, konsultoinnin, tietojärjestelmien kehittämisen ja integroinnin koko tietoliikenteen ja median arvoketjuun.
Yrityksessä X toteutetaan pääosin yhden ison asiakkaan hankkeita. Asiakas on teleoperaattori, jonka omistamaa järjestelmää kehittää ja ylläpitää yritys X. Yksittäisten projektien toteutukseen sisältyy projektinhallinta, suunnittelu, toteutus, testaus ja toimitus asiakkaalle. Yrityksessä toimii seitsemän hengen testausryhmä ja yksi henkilö ryhmästä koordinoi testausta. Olen itse työskennellyt yrityksessä miltei kaksi vuotta testaajana ja olen päässyt tutustumaan testauksen eri ulottuvuuksiin lähietäisyydeltä. Systeemitestauksen ohella tehty regressiotestaus, jolla varmistetaan testausympäristöjen oikea toiminta, on jäänyt tekemättä tai sitä on tehty vähäinen määrä muiden töiden vuoksi. Joulukuussa 2008 yrityksessä otettiin koekäyttöön automaattinen testausväline, CA Automated Regression Testing for CICS, jolla pyrittiin parantamaan regressiotestauksen sujuvuutta. Yrityksestä käytetään tässä opinnäytetyössä nimeä X, koska kyseinen yritys ei halunnut arkaluonteista tietoa julkaistavan suoraan yrityksen nimellä.

# 2 Tavoitteet, menetelmät ja rajaukset

## 2.1 Opinnäytetyön tavoitteet

Tämän opinnäytetyön tavoitteena on tutkia Yrityksen X nykyistä regressiotestausprosessia ja löytää siitä mahdollisia kehitystarpeita. Lisäksi työssä tutkitaan miten automaattinen testaustyökalu, CA Verify Automated Regression Testing for CICS, soveltuu yrityksen regressiotestaustarpeisiin ja lopuksi arvioidaan mahdollisen hankinnan kannattavuutta ajankäytön osalta.

## 2.2 Opinnäytetyön tutkimusmenetelmät

Opinnäytetyön tutkimusmenetelmä oli kvalitatiivinen. Kvalitatiiviselle tutkimukselle on ominaista kokonaisvaltainen tiedon hankinta ja aineiston kokoaminen luonnollisissa, todellisissa tilanteissa. Aineistoa tarkastellaan monipuolisesti ja yksityiskohtaisesti (Induktiivinen analyysi). Tietoa hankitaan muun muassa osallistuvan havainnoinnin keinoin, kyselyin ja haastattelujen avulla. (Hirsjärvi, Remes & Sajavaara 2007, 160.)

Tässä tutkimuksessa menetelminä käytettiin tiedonhakua aiheeseen liittyvistä lähdeaineistoista, sidosryhmien haastatteluita ja havainnointia työn edetessä. Työn aluksi tehtiin yrityksen X testausryhmälle testausautomaatioon ja regressiotestaukseen liittyvä verkkokysely. Lisäksi tutkimusmenetelmänä käytettiin avoimia haastatteluja yrityksen X edustajan sekä CA Incorporatesin suomalaisen edustajan kanssa Verify Automated Regression Testing for CICS –sovellukseen liittyen.

## 2.3 Rajaukset

Termiä regressiotestaus voidaan käyttää monella testaustasolla. Tässä työssä keskitytään regressiotestaukseen järjestelmätasolla, joskin tutkittava työkalu sopii myös muunlaiseen testaukseen.

## 2.4 Tutkimuskysymykset

Tutkimuksen tavoitteena on saada vastaukset seuraaviin kysymyksiin:

- Millainen on regressiotestausprosessin nykyinen toimivuus?

- Millaisia vaikutuksia CA Verify Automated Regression Testing for CICS- testaustyökalulla on yrityksen regressiotestausprosessiin?

# 3 Testaus

Laatu on käsite, josta jokaisella on oma mielikuvansa. Black määrittelee laadun olevan niiden ominaisuuksien ja toimintojen löytymistä tuotteesta, jotka tyydyttävät käyttäjiä ja asiakkaita sekä vastaavat tuotteelle asetettuja vaatimuksia. Se on myös vastaavasti niiden ominaisuuksien puuttumista tuotteesta, jotka eivät tyydytä käyttäjiä eivätkä vastaa vaatimuksia. (Black 2004, 4.) Jotta organisaatio voi varmistaa laadun tuotteessaan, on tuotetta testattava. Seuraavassa käydään läpi ohjelmistojen testausta ja sen suhdetta laatuun.

## 3.1 Testausprosessi ja laadunvarmistus

Haikala ja Märijärvi määrittelevät ohjelmistotuotteen laadun kykynä täyttää käyttäjänsä kohtuulliset toiveet ja odotukset. Näin laatu on käsitteensä subjektiivinen ja käyttäjästä sekä käyttöympäristöstä riippuvainen. Laadun varmistuksessa käytettäviin järjestelmiin on useita malleja. Näistä malleista tärkeimpiä on ISO 9001 -standardi, joka määrittelee tietyt perusasiat, jotka laatujärjestelmän tulee sisältää. Standardin noudattaminen ja laatusertifikaatin hankkiminen eivät välttämättä todista, että laatujärjestelmä on erinomainen, vaan todistavat, että sertifikaatin omistaja toimii laatujärjestelmänsä mukaisesti. Tuotteen laadunvarmistuksen tehtävä on estää virheiden pääsy tuotteeseen ja myös löytää tehdyt virheet mahdollisimman aikaisin. Laadunvarmistusta voidaan tehdä muun muassa testaamalla. (Haikala & Märijärvi 2004, 48-51)

IEEE standardin 610.12 mukaan laadunvarmistus (Quality Assurance, QA) sisältää kaikki tarvittavat toiminnot, joilla saavutetaan riittävä luottamus järjestelmän laatuun. Lisäksi laadunvarmistuksessa arvioidaan prosessia, jolla järjestelmää kehitetään. Testaus koostuu toiminnoista, joilla havaitaan eroavaisuuksia nykyisten ja vaadittujen tilojen välillä. Näin voidaan katsoa, että laadunvarmistus keskittyy kokonaisen prosessin läpiviemiseen oikealla tavalla ja että testaus toimii laadun arvioijana. (IEE 610.12, 1990; Tamres 2002, 217.)

Black toteaa testauksen olevan prosessi, joka arvioi järjestelmän laatua tuottaen samalla palveluita ja tietoa, jotka auttavat organisaatiota hallitsemaan laatuun liittyviä riskejä. Testauksella pyritään löytämään niitä ongelmia, joiden takia yhden tai useamman käyttäjän kohtuulliset odotukset laadusta eivät täyty. Näitä ongelmia kutsutaan bugeiksi tai virheiksi. (Black 2004, 8.)

Fewsterin ja Grahamin mukaan ohjelmistoa on testattava, jotta saavutetaan varmuus, että se toimii sille tarkoitetussa ympäristössä niin kuin pitää. Ohjelmistotestauksen on oltava tehokasta olemassa olevien virheiden löytämisessä, mutta silti testit tulee suorittaa niin nopeasti ja halvalla kuin mahdollista. (Fewster & Graham 1999, 3.) Testaajan tehtävänä on määrittää

toimiiko valmis tuote oikein vertaamalla sitä alkuperäisiin määrittelyihin. Tämän perusteella voidaan päätellä, onko lopullinen järjestelmä valmis julkaistavaksi. Kun testataan järjestelmätasolla, testausympäristön tulee olla mahdollisimman lähellä aiotun ympäristön kaltaista. Jos näin ei ole, testaajan ei voida katsoa arvioivan lopullista tuotetta. (Tamres 2002, 223.)

Farrell-Vinayn (2008, 17-18) mukaan testaus voi tuoda vastauksen erityisesti kahteen kysymykseen:
- Onko tuote valmis julkaisua varten?
- Onko tuotteen testauksessa saavutettu riittävä kattavuus?

Haikala ja Märijärvi toteavat, että testauksen avulla on mahdollista osoittaa, että ohjelmassa on virheitä, mutta ohjelman täyttä virheettömyyttä testauksella ei sen sijaan voida tarkistaa. Testaukseen liittyviin työvaiheisiin ja niihin liittyvään virheiden jäljitykseen ja korjaukseen kuluu tyypillisesti yli puolet ohjelmistoprojektin resursseista. Niinpä testauksen läpivientiin parhaalla mahdollisella tavalla kannattaa kiinnittää huomiota. (Haikala & Märijärvi 2004, 283, 286.)

## 3.2 Testitapaukset

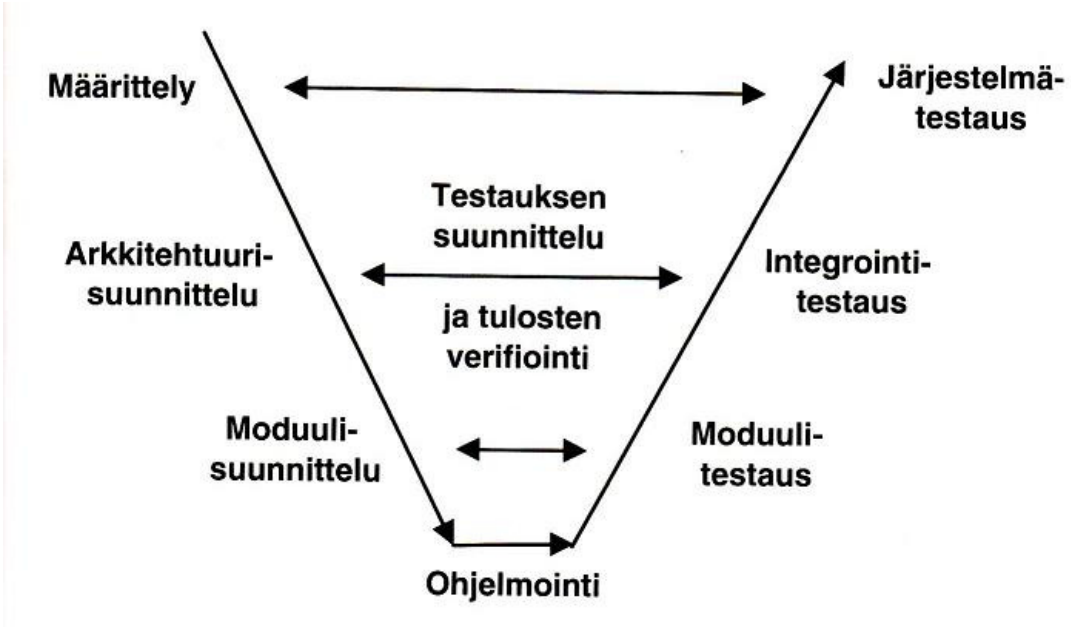
Black (2004, 529) toteaa testitapauksen olevan jono toimintoaskeleita, joita järjestelmässä suoritetaan testin aikana. Fewsterin ja Grahamin (1999, 6) mukaan järjestelmätestauksen testitapaukset on tarkoitettu löytämään virheitä järjestelmän toiminnallisuudesta.

Testitapausten valintaan on kaksi peruslähestymistapaa: lasilaatikkotestaus ja mustalaatikkotestaus. Mustalaatikkotestauksessa testitapaukset valitaan testattavan ohjelman määrittelyjen perusteella tutustumatta ohjelman toteutukseen ja lasilaatikkotestauksessa testitapausten valinnassa käytetään hyväksi tietoja ohjelman toteutuksesta sekä ohjelman koodista. (Haikala & Märijärvi 2004, 291; Black 2004, 25.)

## 3.3 Testauksen V-malli

Ohjelmistotuotannossa käytetään usein niin sanottua V-mallia havainnollistamaan ohjelmiston suunnittelun ja testauksen välistä suhdetta.

Fewsterin ja Grahamin mukaan ohjelmistotuotannon V-malli havainnollistaa, milloin testausta tulisi tehdä. Mallista nähdään että jokaisella suunnittelutehtävällä on vastaava testaustehtävä. (Fewster & Graham 1999, 6.)




Kuvio 1. Testauksen V-Malli. (Haikala & Märijärvi 2004, 289.)

Haikalan ja Märijärven (2004, 288) mukaan järjestelmätestausta voi joskus seurata erillinen kenttätestaus tai hyväksymistestaus. Fewster ja Graham korostavat että V-mallin onnistuneen hyödyntämisen taustalla on testitapausten suunnittelun ajankohta. Testien suunnittelutoiminnot löytävät virheitä testauskohteistaan sitä tehokkaammin mitä aikaisemmin ne aloitetaan. (Fewster & Graham 1999, 6.)

Seuraavassa kuvataan tarkemmin V-mallin testaustehtäviä:

**”Moduuli- eli yksikkötestauksessa** testattavana on yksittäinen moduuli. Moduuli koostuu yleensä noin 100-1000 ohjelmarivistä. Moduulin toimintaa verrataan moduulisuunnittelun ja arkkitehtuurisuunnittelun tuloksiin, tavallisimmin tekniseen määrittelydokumenttiin. Testauksen suorittaa yleensä moduulin toteuttaja.” (Haikala & Märijärvi 2004, 289.) Yksikkötestauksen testitapaukset on tarkoitettu löytämään virheitä koodista (Fewster & Graham 1999, 6).

**Integrointitestaus** on testausvaihe, joka on keskittynyt järjestelmän komponenttien ja alijärjestelmien suhteiden ja rajapintojen testaamiseen. (Black 2004, 524.) Integrointitestauksen testitapaukset on tarkoitettu löytämään virheitä arkkitehtuurista (Fewster & Graham 1999, 6).

**Järjestelmä- eli systeemitestauksessa** tarkastellaan kokonaista järjestelmää ja tuloksia verrataan määrittelydokumentaatioon. Järjestelmätestauksessa testataan myös järjestelmän ei-toiminnalliset ominaisuudet kuten kuormitustestit, luotettavuustestit ja asennustestit. (Haikala & Märijärvi 2004, 290.)

Haikala ja Märijärvi toteavat että mitä korkeammalla V-mallin testaustasolla ollaan, sen kalliimmaksi virheiden korjaus tulee. Virheiden korjaus voi myös aiheuttaa uusia virheitä. Kun esimerkiksi järjestelmätestauksessa havaittu virhe korjataan, voi korjaus aiheuttaa muutoksia useisiin moduuleihin. Siltä varalta, että jokin muutostarve jää huomaamatta, myös muut moduulit pitäisi testata ja lopuksi vielä suorittaa järjestelmätestaus (mieluiten kokonaan) uudelleen. Tällaista uudelleentestausta (testaustasosta huolimatta) kutsutaan regressiotestaukseksi. (Haikala & Märijärvi 2004, 290.)

Regressiotestaus, jota tässä opinnäytetyössä pääosin käsitellään, on myös V-mallin yhteydessä käytetty termi. (Haikala & Märijärvi 2004, 288.) Se toteutetaan järjestelmätestauksen menetelmin ja on tärkeä osa toimivaa testausprosessia.

## 3.4 Regressiotestaus

Tamres määrittelee regressiotestauksen koostuvan aikaisemmin ajetun testisarjan ajamista uusilla versioilla sovelluksesta. Tavoite on varmistaa, että toiminnot, jotka toimivat edellisissä versioissa toimivat edelleen odotetusti. Usein virheen korjaaminen tai uuden ominaisuuden lisääminen saattaa rikkoa jotain muuta, joka on ennen toiminut. (Tamres 2002, 223.)

Regressiotestauksen tehtävänä on tarkistaa, että muutokset ohjelmistossa tai ympäristössä eivät ole aiheuttaneet ei-haluttuja sivuvaikutuksia ja että järjestelmä on edelleen määritysten mukainen (Grove Consultants 2001, 10). Regressiotestauksella pyritään löytämään bugeja, jotka aiheuttavat regressiota. Järjestelmässä esiintyy regressiota, kun muutoksen tuloksena järjestelmän versio n+1 sisältää bugin, jota ei ollut versioissa 1-n. Tämä aiheuttaa aikaisemmin oikein toimineen järjestelmän osan käyttäytymään väärin. (Black 2004, 177.) Regressiotestausta tulisi tehdä jokaisen muutoksen jälkeen, joka saattaa vaikuttaa vanhaan toimivaksi todettuun ohjelmaan (Dustin 2003, 201). Myös Tamresin (2002, 223) mukaan ihanteellinen testiympäristö on sellainen, jossa testaaja suorittaa regressiotestit jokaisella kerralla kun sovellus muuttuu. (Tamres 2002, 223.)

Gaon mukaan yksi regressiotestauksen suurimpia ongelmia on järjestelmän määritysten muutosten tunnistaminen järjestelmällisellä tavalla tai työkalulla. Vastaavasti on tunnistettava miten nämä määritysten muutokset vaikuttavat järjestelmään. Näiden tietojen perusteella on rakennettava ja muokattava testitapaukset kattamaan riittävä alue, jotta uudelleentestausta

tarvittaisiin mahdollisimman vähän. (Gao 2002.) Dustin toteaa, että vaikka regressiotestaus on todettu tärkeäksi, se jää usein vähimmälle huomiolle testitoimintoja suunnitellessa. Suunnittelematon ja manuaalinen lähestymistapa voi johtaa tehottomaan ja riittämättömään regressiotestaukseen, jossa resursseja ei käytetä tehokkaasti. Regressiotestauksen tulisi keskittyä suurimman riskin omaavaan toiminnallisuuteen. Kun tämä toiminnallisuus on testattu, voidaan tutkia yksityiskohtaisemmin muita toimintoja. (Dustin 2003, 201.) Tamresin mukaan sovellus ei ole virheetön vaikka joukko regressiotestejä menisi onnistuneesti läpi. Uusia testejä tarvitaan varmentamaan äskettäin lisättyjä ominaisuuksia. (Tamres 2002, 223.)

# 4 Testausautomaatio

Viime vuosina työkalut sekä menetelmät sovellusten ja käyttöliittymien ohjelmoinnissa ovat kehittyneet suuresti parantaen ohjelmoinnin tehokkuutta. Tämä on asettanut lisäpaineita testaajille, jotka mielletään usein ohjelmistotuotteiden toimituksen pullonkauloiksi. Testaajilta vaaditaan yhä parempaa tehokkuutta ja tuottavuutta vähemmässä ajassa. Yksi ratkaisu tähän ongelmaan voi olla testauksen automatisointi.

## 4.1 Tarpeiden tunnistus

Fewster ja Graham nimeävät muutamia ongelmia joihin automaatio saattaa tuoda ratkaisun:

- Manuaalisen testauksen ongelmat (Esimerkiksi vie liikaa aikaa, on tylsää, virhealtista)
- Ei aikaa regressiotestaukselle, kun ohjelmiin tehdään pieniä muutoksia
- Testidatan tai testitapausten valmistelu virhealtista
- Riittämätön testidokumentaatio
- Tietämättömyys siitä, paljonko ohjelmasta on testattu
- Testaus on tehotonta

Kirjoittajien mukaan testauksen ongelmat kannattaa laittaa järjestykseen ja tarttua vakavimpiin ongelmiin suurimman edun saavuttamiseksi. (Fewster & Graham 1999, 252-254.)

Dustinin mukaan oltaessa tekemisissä laajan ja monimutkaisen järjestelmän kanssa, voidaan päätyä tuhansiin regressiotesteihin, kun edellisten versioiden testisetit lisätään uuteen settiin

mukaan. Tämän takia on tarpeellista, että regressiotestisarjat automatisoidaan ja ajetaan vakaassa testiympäristössä, joka mahdollistaa kaikkien testien ajamista nopeasti käyttämättä suuria määriä resursseja. Jos näitä testejä ei automatisoida, regressiotestaus saattaa muuttua pitkäksi ja pitkäveteiseksi prosessiksi. Pahimmillaan jotkut regressiotestit saattavat jäädä tekemättä jättäen samalla suuria aukkoja testauksen tavoitteeseen. Lisäksi regressiotestien manuaalinen suorittaminen voi olla tylsää, virhealtista ja kattamatonta. (Dustin 2003, 203.)

## 4.2 Automatisoinnin edut

Automatisoimalla saavutetaan kattavampi testaustulos, sama testaustulos jokaisella testauskierroksella ja mahdollisuus testata vuorokauden ympäri. Lisäksi onnistunut automatisointi vapauttaa manuaalisia testaajia kokeelliseen tai muuhun vaativampaan testaukseen. Automaatio tuottaa myös suorituskyky- ja kestävyystietoa ohjelmasta. Tärkein ominaisuus lienee kuitenkin ajankäytössä saavutetut kustannussäästöt sekä laajennetut testausmahdollisuudet. (Hinkkanen & Leiman 2008, 23-24; Pyhäjärvi & Pöyhönen 2004, 16; Farrell-Vinay 2008, 97.)

Tamresin (Tamres 2002, 225.) mukaan testauksen automatisointi voi lisäksi tuoda organisaatiolle sellaisia pitkän aikavälin etuja kuten

- testaajan osuuden väheneminen testien suorittamisessa
- satojen käyttäjien simuloimisen mahdollistuminen
- inhimillisiltä virheiltä välttyminen, työkalut hallitsevat toistuvat ja tylsät testit

## 4.3 Automatisoinnin haasteita ja riskejä

Testausautomaatio nostaa odotuksia, mutta usein turhauttaa ja pettää käyttäjät. Vaikka automaatio lupaa tien ulos vaikeasta tilanteesta, automaattisten testien suorittaminen saattaa luoda yhtä monia ongelmia kuin mitä se ratkaisee. Testausautomaation tavoite ja haaste on vähentää manuaalisesti tehtävää testausta, ei poistaa sitä kokonaan. (Pettichord 2001a; Pettichord 2001b; Pyhäjärvi 2006.) Automaatio ei ole myöskään ole ainoa tapa kehittää testausprosessia eikä kaikkia kannata automatisoida. Hinkkasen ja Leimanin (2008, 14) mukaan tällaisia testejä ovat esimerkiksi yksinkertaiset testit, keskeneräisten tuotteiden testit ja liian monimutkaiset testit. Pyhäjärvi ja Pöyhönen toteavatkin, että automatisoinnin edut voidaan saavuttaa myös muilla prosessin kehittämisen välineillä. Hyödyt suhteessa kustannuksiin ovat tärkeä vaatimus välineen käytölle. Jos tarvittavaa hyötyä ei saavuteta, kustannukset voivat nousta ennakoimattoman suuriksi. (Pyhäjärvi & Pöyhönen, 2004.)

Tamresin mukaan automaatio ei ole ilmaista, vaan hyödyn saavuttamiseksi on tehtävä huomattava määrä työtä. Pelkkä testaustyökalun hankkiminen ei automatisoi testausprosessia

onnistuneesti. Hyödyllisen automaattisen testausympäristön rakentaminen vaatii aikaa ja kustannushyötyjä nähdään vasta useiden testien toistojen jälkeen. Testejä ei tulisi automatisoida jos niissä vaaditaan manuaalista toimintaa tai ne aiotaan ajaa vain kerran tai erittäin harvoin. Kirjoittajan mukaan seuraavat seikat liittyvät testausautomaation onnistumiseen:

- Automaattiset testaustyökalut eivät määrittele tai luo testejä. Testien luomiseen tarvitaan huomattava määrä aikaa.
- Testiskriptit itsessään ovat ohjelmointikieltä. Testien suorittamisen automatisointi saattaa vaatia ohjelmointikokemusta.
- Testityökalut eivät valitse oikeita testejä automatisoitaviksi. Se pitää tehdä itse.
- Testaajat tarvitsevat koulutusta työkalujen käyttöön.
- Millä tahansa uudella työkalulla on jyrkkä oppimiskäyrä.
- Työkalut vaativat rakenteen ylläpitoa jotta tiedostoja ja testeihin liittyviä luomuksia voidaan hallita.
- Automatisoitu testi joka ei löydä virheitä ei takaa, ettei muita virheitä ole.
- Testisarjojen ylläpito saattaa vaatia paljon aikaa ja vaivaa.
- Uusia testejä tarvitaan aina koska testit kuluvat ajan myötä. Olemassa olevat testit ovat jo löytäneet virheet joita niiden oli tarkoitus löytää.
- Testityökalut ovat myös ohjelmistoja, joten ne voivat sisältää bugeja.

(Tamres 2002, 226.)

Automatisoitua testausta voidaan mitata tarkkailemalla ajankäyttöä ja työmääriä täsmällisesti; myös automaation ylläpitoon menevä aikaa tulee huomioida. Löytyneiden virheiden määrää ja mahdollisia suorituskykymuutoksia tulee myös pitää silmällä. (Hinkkanen & Leiman 2008, 25-26; Pyhäjärvi 2006.)

### 4.4 Testitapaukset automaatiossa

Pettichordin mukaan on tärkeää, että automaatiossa käytettävään testitapauksien sarjaan, testisettiin, voidaan luottaa. Sen tulee olla kykenevä ja tarkka toimimaan järjestelmän uusissa versioissa. Vakavin asia, joka tällaiselle testisetille voi tapahtua, on se että ohjelma ilmoittaa testien menneen läpi vaikka todellisuudessa toiminnoissa, joita on tarkoitus testata, saattaa olla ongelmia. (Pettichord, 2001b.) Mitä enemmän testitapauksia yrityksen testipakettiin kuuluu, sitä enemmän testejä on ylläpidettävänä. Mitä enemmän testejä on ylläpidettävänä, sitä enemmän ylläpitoon kuluu aikaa ja vaivaa. Fewster ja Graham (1999, 192-193) kehottavatkin karsimaan testipaketista päällekkäisiä ja epäolennaisia testitapauksia.

Fewsterin ja Grahamin mukaan useimmat testitapaukset vaativat joidenkin ennakkoehtojen toteuttamista ennen kuin testien suorittaminen voi alkaa. Nämä ehdot tulisi määritellä jokai-

seen testitapaukseen ja toteuttaa ne ennen testin tekemistä. Testit voivat tarvita esimerkiksi tietokannan, jossa on tiettyjä asiakastiedostoja valmiina. Jotkin testitapaukset vaativat palautuksen jokaisen suorituskerran jälkeen, koska ennakkoehdot muuttuvat testitapauksen suorittamisen aikana. Fewster ja Graham käyttävät menettelystä nimeä esikäsittely. Esikäsittelyn tekeminen manuaalisesti on virhealtista ja aikaa vievää. Jos halutaan automatisoida testaus, myös esikäsittelyt on automatisoitava. Kun automatisoitu ympäristö on saatu valmiiksi regressiotestausta varten, voidaan regressiotestejä ajaa huomattavasti pienemmässä ajassa verrattuna käsin tehtäviin testeihin. (Fewster & Graham 1999, 176-178, 254.)

## 4.5 Oikean testaustyökalun valitseminen

Hinkkasen ja Leimanin (Hinkkanen & Leiman 2008, 39) mukaan testaustyökalua valitessa tulee ottaa huomioon tuotteen seuraavat ominaisuudet:

- Kattavuus
- Luotettavuus
- Kapasiteetti (Minkä suuruisia tietomääriä työkalu pystyy käsittelemään)
- Opittavuus (Koulutustarjonta ja materiaalin riittävyys)
- Käytettävyys (Kuinka helppo työkalu on käyttää)
- Suorituskyky (Onko työkalu tarpeeksi nopea)
- Yhteensopivuus (Toimiiko tuote yrityksen ympäristössä)

Tamresin mukaan testausautomaatio koskee kahta avainasemassa olevaa testaustapahtumaa: testien suorittamista ja tuloksen arvioimista. Monet kaupalliset työkalut tukevat näitä tapahtumia. Jotkut yleisimmistä työkaluista sisältävät seuraavia ominaisuuksia:

**Nauhoitus/Toisto -työkalut** nauhoittavat tapahtumia (mukaan lukien näppäinten painaminen, hiiren liikkeet ja näytön tuloste) samalla kun testaaja ajaa sovellusta, ja asettaa tiedon scriptiin eli komentokielellä tehtyyn lauseeseen. Teoriassa työkalu voi sitten ajaa nauhoituksen uudelleen testatakseen sovellusta. Käytännössä, tallennettuja testejä voi käyttää rajoitetusti, mutta hyvin muodostetut suoritusscriptit antavat merkittäviä etuja.
**Vertaavat työkalut** etsivät eroavaisuuksia kahdesta tiedostosta ja määrittelevät onnistuiko testi vai ei. Testaaja voi usein määrittää osan tiedosta ohitettavaksi. Esimerkiksi päivämäärä/aika –kentän eroavaisuuden ohittaminen johtaa hyväksyttyyn testiin jos muu osa tiedosta on oikeaa. **Testien suoritustyökalut** muodostavat ympäristön joka alustaa sovelluksen, lähettää tiedon sovellukseen, tallentaa tulosteet, lähettää tulosteet vertaajaan arviointia varten ja kirjaa tulokset lokille. (Tamres 2002, 225-226.)

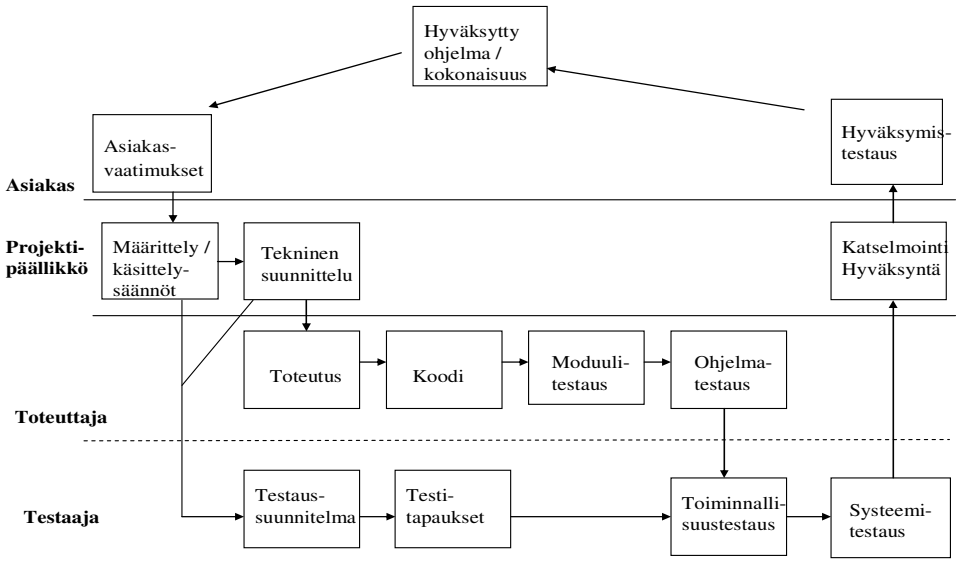
# 5 Testausprosessi yrityksessä X

Tässä luvussa kuvataan, miten testaus on järjestetty Yrityksessä X ja miten testausta käytännössä tehdään. Erityisesti kappaleessa keskitytään yrityksen regressiotestausprosessiin ja sen tutkimiseen.

## 5.1 Testausprosessin tarkoitus

Yrityksen X käsikirjassa yrityksen testausprosessia kuvataan seuraavasti: Testausprosessin tarkoitus on varmistaa, että asiakas saa määritysten mukaisesti toimivan tuotteen ja että testaus suoritetaan ohjatusti ja toistettavasti. Testausprosessi ohjaa oikea-aikaiseen testaamiseen sovelluksen eri kehitysvaiheissa. Mahdollisimman varhaisessa vaiheessa havaittujen poikkeaminen ja puutteiden korjaaminen on helpompaa ja halvempaa kuin niiden korjaaminen sovelluksen toteutuksen valmistuttua. Ennakoivia eli ennen toteutustyön aloittamista käytettyjä testaustapoja ovat vaatimusmäärittelyyn, toiminnallisen määrittelyyn ja tekniseen suunnitteluun liittyvien dokumenttien tarkastus ja katselmointi. Testaamisen tavoitteena on varmistaa ohjelmistotuotteen laatutaso löytämällä virheitä niin aikaisessa vaiheessa kuin mahdollista. Testaamisella pyritään löytämään ja korjaamaan valtaosa virheistä niin, että asiakkaan löytämät virheet minimoituvat lukumäärältään ja vakavuudeltaan. Tuotteen tai toiminnallisuuden lisääminen edellyttää myös, että tuotteen toimivuuden lisäksi varmistetaan järjestelmän perustoimintojen toimivuus uudessa tilanteessa eli suoritetaan ns. regressiotestausta. (Yritys X, 2007.)

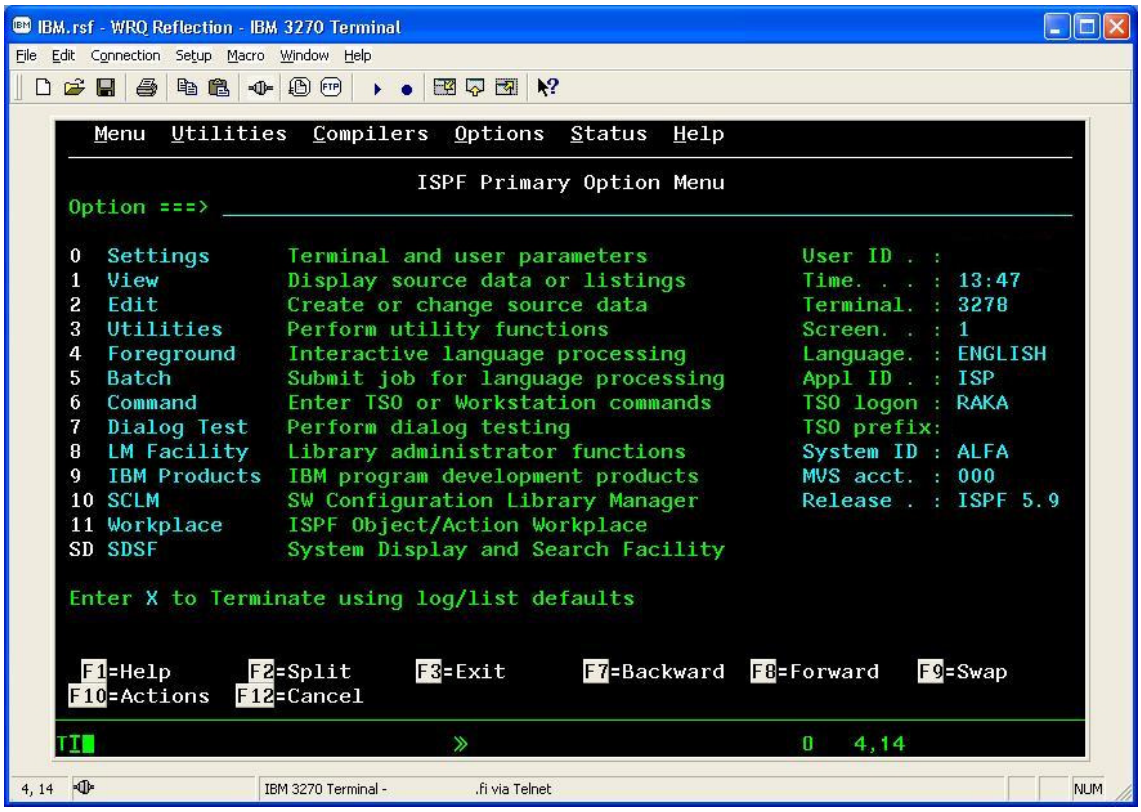
Projektiprosessi johtaa testaukseen, kun projektipäällikkö varaa koodaajan ja testaajan projektiinsa Yrityksen X sisäisellä resurssinhallintasovelluksella. Testausprosessi alkaa, kun koodaaja toteuttaa ja testaa yleensä itse koodaamiaan palasia ohjelmasta eli suorittaa moduulitestausta. Samaan aikaan testaaja tekee testaussuunnitelman projektin määrittelyjen perusteella ja sen valmistuttua suunnitelma katselmoidaan asiakkaan kanssa. Tämän jälkeen testaussuunnitelmaa tarkennetaan testitapauksiin. Kun moduulit on testattu yksitellen, toteuttaja testaa niitä yhdessä kokonaisena ohjelmana. Mikäli ohjelma näyttää toimivan, ohjelman toteuttaja siirtää ohjelman systeemitestiympäristöön, jotta testaaja voi aloittaa ohjelman testaamisen tarkennetun testaussuunnitelman ja testitapausten mukaisesti. Testaaja raportoi testatessa ilmenneet virheet projektipäällikölle sekä toteuttajalle. Korjauksen jälkeen virheet testataan uudelleen ja kun virhe on todettu korjatuksi, se suljetaan. Kun testaussuunnitelman mukaiset testit on tehty ja mahdolliset virheet ovat korjattu onnistuneesti, katselmoidaan testitulokset asiakkaan kanssa. Mikäli asiakas toteaa katselmointitilaisuudessa tulosten vastaavan määrittelyjä, voidaan ohjelma siirtää eteenpäin tuotantoon ja asiakkaan käyttöön. Asiakas saattaa tehdä hyväksymistestausta sille tarkoitetussa ympäristössä järjestelmätestauksen loppuvaiheessa tai sen valmistuttua.




Kuva 2. Yrityksen X projektiprosessi.

## 5.2 Testauskäytännöt- ja työkalut

Yrityksessä käytössä oleva järjestelmä on merkkipohjainen, 1980-luvulla kehitetty asiakastietojärjestelmä. Järjestelmässä on tuotanto-, pilotti- ja systeemitestitietokannat. Testaaminen tapahtuu pääosin Mainframe -koneella sijaitsevissa systeemitestiympäristöissä. Mainframe on erittäin suorituskykyinen suuri tietokone, joka tukee tuhansia sovelluksia ja sähköisiä laitteita palvellakseen samanaikaisesti suuria määriä käyttäjiä. Suurkonetta käytetään varastona tietokannoille, tapahtumanhallintapalvelimille ja sovelluksille, jotka vaativat korkean tason tietoturvaa ja käytettävyyttä. Mainframe on tässä tapauksessa IBM:n suurkone, johon testaaja ottaa yhteyden WRQ Reflection for IBM- pääte-emulaattorilla. WRQ Reflection mahdollistaa yhteyden Windows -käyttöjärjestelmästä IBM mainframe -koneelle. Emulaattori jäljittelee 3270 -yhteyttä, jolloin suurkone luulee olevansa yhteydessä 3270 -tyyppisen päätteen kanssa. Järjestelmään kirjaudutaan sisään käyttäjätunnuksella ja salasanalla. Testiympäristöihin päästään TPX-valikon kautta, joka sisältää käyttäjän käyttöoikeuksien puitteissa määritellyt mahdolliset istunnot. TPX (Terminal Productivity Executive) – sessionhallintaohjelmalla voidaan pitää samanaikasesti auki useita eri virtuaali-istuntoja ja hyppiä niiden välillä kirjautumatta ulos välillä.




Kuva 3. Esimerkki emuloidusta 3270-yhteydestä.

Yrityksessä on vuoden 2008 aikana otettu käyttöön Mercury Quality Center- työkalu. Se on web-pohjainen testauksen hallintaan ja dokumentointiin tarkoitettu sovellus. Ennen käyttöönottoa testaussuunnitelmat ja testitapaukset dokumentoitiin Microsoft Office Word- ohjelmalla verkkokansioihin. Nykyään käytäntönä on, että testaussuunnitelmat ja testitapaukset tallennetaan Quality Centeriin. Sillä voidaan hallita muun muassa määrittelyjä, testitapauksia- ja tuloksia sekä virheraportointia. Jokaiseen suoritettuun testiin voidaan tallentaa esimerkiksi testin suorittaja, ajankohta ja lopputulos. Näin Quality Center toimii myös testauspäiväkirjana.

## 5.3 Testausympäristöt

Systeemitestiympäristöissä, joita on kaikkiaan viisi kappaletta, on mahdollista suorittaa systeemitestit yhdessä ympäristössä häiritsemättä neljää muuta. Ympäristöt sijaitsevat IBM-suurkoneella. Systeemitestiympäristöt ovat mahdollisimman kattavasti tuotannon kaltaisia ja niissä on mahdollista ajaa kaikkia tuotannossa automaattitoiminnoilla hallittuja eräajoja. Ympäristöillä on keskenään toisistaan riippumattomat, erilliset CICSit ja tietokannat. CICS eli Customer Information Control System on IBM:n kehittämä On-line tapahtumien hallinnointijärjestelmä, joka on käytössä monilla suurilla monikansallisilla yrityksillä. Sitä käytetään esimer-

kiksi pankkiautomaateissa, elektronisessa osakekaupassa ja elekronisten lippujen hallinnassa. (IBM, 2004.) Ympäristöt käyttävät yhteisiä ohjelmakirjastoja. Tästä seuraa se, että systeemitestiin siirretyt ohjelmat ovat testattavissa jokaisessa ympäristössä. Testiympäristöjen ylläpitoon on nimetty henkilö, ja testaajat ylläpitävät kuukausittain tehtävää perusversiota. Perusversio on varmuuskopio tietokantatilanteesta, johon on viety uusimmat ohjelmamuutokset ja sen tulee olla ajan tasalla muidenkin asioiden, kuten testiaineiston, suhteen. (Yritys X, 2008.)

## 5.4 Regressiotestaus

Regressiotestauksen tarkoituksena on varmistaa, että uuden tuotteen tai toiminnallisuuden lisäämisen jälkeenkin järjestelmä toimii määritellyllä tavalla. Regressiotestaus on jatkuvaa toimintaa laadun varmistamiseksi. (Yritys X, 2007). Eniten muutoksia yrityksen järjestelmässä tapahtuu online- toiminnoissa kuten hinnoittelussa ja tuotteiden nimissä. Testitapaukset ovat keskittyneet online- toimintojen testaamiseen ja regressiotestaus onkin kulkenut yrityksessä nimellä ”Online- hinnoittelun regressiotestaus”.

### 5.4.1 Käytännöt

Regressiotestausta on tehty systeemitestauksen menetelmin tietokannassa, jossa on tuotannon ohjelmaversiot online-hinnoittelun ohjelmista. Näin saadaan jäljiteltyä sovellukselle tarkoitettua ympäristöä mahdollisimman tarkasti. Regressiotestikierros vie 3-4 päivää yhdeltä testaajalta, mutta resursseja on priorisoitava projekteihin, joten regressiotestausta ei ole ehditty tehdä haluttua määrää. Testitapausten päivitystä ei ole tehty säännöllisesti, joten voidaan olettaa että regressiotestauskierroksen suorittamiseen kuluu nykytilanteessa vielä enemmän aikaa. Vuonna 2008 regressiotestauskierros tehtiin vain muutaman kerran. Muutoksia järjestelmään tulee monta kertaa vuodessa, usein jopa kuukausittain. Yrityksessä X on käytössä ns. Release- malli, jossa viedään samaan aikaan tuotantoon useita asiakkaan projekteja. Tällöin myös useisiin ohjelmiin tulee samanaikaisesti muutoksia, jotka saattavat vaikuttaa koko järjestelmään. Testidokumentaatio on osittain puutteellista, koska regressiotestitapaukset eivät ole täysin ajan tasalla. Järjestelmä muuttuu usein, joka asettaa haasteita testitapausten suunnittelulle jatkuvuutta ajatellen. Regressiotestaukseen käytettäviä testitapauksia ei ole viety Quality Center- työkaluun. Testitapausten vieminen Quality Centeriin ja päiväkirjan muotoon helpottaisi regressiotestauksen hallintaa jatkossa.

Ennen regressiotestauksen aloittamista testauksessa käytettävään systeemitestiympäristöön on vietävä ajan tasalla oleva ja uusimmat ohjelmamuutokset sisältävä varmuuskopio.
Tämän jälkeen on huolehdittava, että kaikki on myös muuten valmista regressiotestikierrosta varten. Käytännössä tämä tarkoittaa testidatan muuttamista systeemitestiympäristöön ennen

testien aloittamista. Yrityksessä X kaikilla testaajilla on oikeudet tehdä tarvittavat muutokset tietokantaan ennen testien aloittamista.

Kun regressiotestikierros on ajettu, on testeissä tehdyt päivitykset kirjoitettu tietokantaan. Jos testikierros halutaan ajaa uudelleen, on testiympäristöön palautettava taas regressiotestaukseen soveltuva alkutilanne. Palautus tapahtuu JCL (Job Control Language) eräajolla. Jos regressiotestauksessa löytyy virheitä, ne raportoidaan testauksen esimiehelle, joka ohjaa ne eteenpäin korjausta varten. Yrityksen nykykäytäntö virheraportoinneissa on, että projektien testitapauksista löytyneet virheet raportoidaan Mercury Quality Center- sovelluksessa. Regressiotestaukseen käytettäviä testitapauksia ei ole kuitenkaan vielä siirretty Quality Centeriin ja virheraportointi hoidetaan vanhan kaavan mukaan.

### 5.4.2 Testitapaukset

Grove Consultantsin (2001, 10) mukaan on yleistä, että organisaatioilla on regressiotestipaketti, joka on joukko testitapauksia erityisesti regressiotestausta varten. Myös Yrityksessä X on luotu regressiotestaukseen tarkoitettu Microsoft Word –dokumentti, joka sisältää regressiotestikierroksella ajettavat testitapaukset. Testitapauksia on noin 30 ja jokainen tapaus sisältää useamman askeleen. Testitapaukset on luotu vastaamaan tuotannon määrittelyjä mahdollisimman tarkasti. Kun uusi projekti valmistuu ja se viedään tuotantoon, tulisi sen aiheuttamat muutokset huomioida regressiotestauksen testitapauksissa. Testidokumenttiin merkitään testattaessa testin ajankohta, tulos ja testaaja. Testitulokset dokumentoidaan testattaessa sovelluksesta kuvankaappauksina ja näytön kuvaa verrataan edellisellä testauskerralla saatuun kuvaan. Regressiotestauksen testitapaukset sisältävä dokumentti ei kuitenkaan ole ajan tasalla; sisällysluettelo ei vastaa sisältöä täysin ja testitapaukset ovat vanhentuneet. Väliin on jouduttu lisäämään poikkeustapauksia, jotka ovat ilmenneet testatessa, mutta niitä ei ole otsikoitu oikein tai lisätty sisällysluetteloon. Regressiotestaukseen tarkoitetut testitapaukset saattavat toimia nykyhetkenä eri tavalla kuin tapausten luomishetkellä.

### 5.4.3 Kehitystarpeet

Yrityksen X regressiotestausprosessin parissa työskentelevälle testausryhmälle tehtiin verkkokysely (Liite 1), jossa pyrittiin kartoittamaan prosessin nykytilaa yrityksessä. Kyselyyn vastasi odotetusti kahdeksan henkilöä. Vastanneista henkilöistä 6 on naisia ja 2 miehiä. Kysymyksiin vastattiin seuraavasti:

1. Kuinka tärkeänä pidät regressiotestausta testausympäristön eheyden kannalta? Vastaa asteikolla 4 (ei lainkaan tärkeää) - 10 (erittäin tärkeää)

   Arvosanan 8 antoi 3 henkilöä
   Arvosanan 9 antoi 3 henkilöä
   Arvosanan 10 antoi 2 henkilöä

2. Onko regressiotestausta mielestäsi tehty yrityksessä tarpeeksi usein? Valitse vaihtoehdoista Kyllä, Ei ja En osaa sanoa.

   Vaihtoehdon ’Ei’ valitsi 8 henkilöä

3. Uskotko että testausautomaation käyttäminen regressiotestauksessa parantaisi prosessia? Valitse vaihtoehdoista Kyllä, Ei ja En osaa sanoa.

   Vaihtoehdon ’Kyllä’ valitsi 7 henkilöä
   Vaihtoehdon ’Ei’ valitsi 1 henkilö

4. Anna kouluarvosana 4 (huono) - 10 (erinomainen) regressiotestauksen nykytilalle.

   Arvosanan 5 antoi 3 henkilöä
   Arvosanan 6 antoi 4 henkilöä
   Arvosanan 9 antoi 1 henkilö

Kyselyn tuloksista voidaan päätellä, että enemmistö pitää regressiotestausta melko tärkeänä. Kaikkien vastanneiden mielestä regressiotestausta ei ole tehty yrityksessä tarpeeksi. Suurin osa uskoi testausautomaation parantavan regressiotestausta. Regressiotestauksen nykytilaa pidettiin melko huonona.

Regressiotestaukseen liittyviä kehitystarpeita huomattiin kyselyn tuloksista sekä kirjallisuuteen perustuvissa tutkimuksissa. Oman kokemukseni perusteella suuri ongelma regressiotestauksessa on ajan puute, jonka testauksen automatisointi voisi välillisesti ratkaista, koska sen avulla testit voidaan suorittaa nopeammin. Myös lähdekirjallisuudessa todettiin ajan puutteen olevan suurin syy regressiotestauksen automatisointiin. Näihin haasteisiin haettiin vastausta testausautomaatiosta ja CA Automated Regression Testing for CICS – työkalun koekäytöstä.

# 6 Ca verify automated regression testing for cics koekäyttö

Tässä luvussa kuvataan CA Verify Automated Regression Testing for CICS- testaustyökalun koekäyttöä Yrityksessä X sekä tarkastellaan koekäytön tuloksia.

## 6.1 Ennen koekäyttöä huomioitavaa

Fewster ja Graham suosittelevat kokeilemaan työkalua ensin jonkin pienen pilottiprojektin kanssa. Näin varmistetaan, että mahdolliset kohdatut ongelmat huomataan silloin, kun vain pieni joukko ihmisiä käyttää työkalua. Näin voidaan myös osoittaa, miten työkalu vaikuttaa testaustapoihin ja antaa alustavaa tietoa siitä, miten mahdollisesti nykyistä prosessia joudutaan muokkaamaan, jotta saataisiin suurin hyöty irti työkalusta. (Fewster & Graham 1999, 290.)

Pyhäjärven ja Pöyhösen mukaan työkalun käyttöönoton tehtäviä ovat:

- Rakenteiden luominen
- Pilotointi
- Laajentaminen useisiin käyttäjiin ja käyttöihin
- Ymmärrys että käyttöönoton jälkeen varsinainen työ vasta alkaa

(Pyhäjärvi & Pöyhönen 2004, 27.)

Koekäytön aikana tulee tarkkailla myös testausautomaation vaikutuksia, jotka saattavat olla vaikeasti mitattavissa. Pyhäjärven (2006) mukaan tällaisia ovat mm. seuraavat:

- Testausorganisaation ammattimaisuus
- Testausorganisaation ulkopuolelta koettu tuottavuus
- Laajentuminen edistyneisiin testausasioihin
- Testien laatu
- Halukkuus kokeilla ja saada aikaan muutoksia osalla testausryhmää
- Luottamus testaajien ja johdon välillä
- Yrityksen kyky saada versioita nopeasti läpi testauksesta
- Testausryhmän kyky käyttää vapautunut aikansa tutkivaan testaukseen

Fewsterin ja Grahamin mukaan on tärkeää, että automaattisesti ajetusta testisetistä saadaan raportti ulos. Näin säästetään aikaa ja nähdään testien lopputulos nopeasti. Raportin tulisi sisältää seuraavat tiedot: ajettavien testien määrä, todellisuudessa ajettujen testien määrä, läpimenneiden testien määrä, virheeseen päätyneiden testien määrä, odotettujen virheiden määrä, kesken jääneiden testien määrä, selitykset kesken jääneiden testien virheistä, käyte-

tyt resurssit (koneteho jne.), käytetty aika ja arvioitu aika testien valmistumiseen. (Fewster & Graham 1999, 245.)

## 6.2 Alkutilanne

Tutkimusta aloitettaessa Yrityksessä X ei ollut säännöllisessä käytössä automaattista testaustyökalua. Vuosina 2001-2004 yrityksellä oli ollut käytössä Talc2000- testaustyökalu, joka hankittiin aikanaan helpottamaan regressiotestausta. Keväällä 2008 Talc2000- työkalun käytöstä kerättiin kokemuksia sitä käyttäneiltä testaajilta. Ongelmia todettiin yhteensopivuuden, luotettavuuden ja ylläpidon kanssa. Talc2000 vaati oman työaseman ja yhteyden Australiaan. Lisäksi synkronointi testauksessa käytettävän pääte-emulaattorin, Reflectionin, kanssa oli epävarmaa. Testejä suoritettaessa yksittäinen testi saattoi joskus mennä läpi ja joskus ei. Työkalu ei myöskään löytänyt todellisia virheitä. Ylläpidossa huomattiin, että yrityksen muuttuvassa järjestelmässä testitapauksien ylläpito oli vaikeaa, koska tapaukset piti nauhoittaa aina uudelleen, kun järjestelmään tehtiin muutoksia. Talc2000- testaustyökalusta luovuttiin pääosin siksi, että manuaalisesti testit saatiin ajettua luotettavammin läpi. Talc2000 oli hankittu, mutta siitä ei ikinä saatu varsinaista hyötyä irti. Yrityksessä X CA Verify Automated Regression Testing for CICS- työkalun haasteiksi nähdään työkalun hyöty suhteessa kustannuksiin, käytettävyys ja resurssien nimeäminen ylläpitoon. Jos työkalu hankitaan, on siitä saatava todellista hyötyä.

Edellä mainitulla tilanteella oli myös vaikutusta päätökseen ottaa CA Verify Automated Regression Testing for CICS- työkalu koekäyttöön. Yrityksessä X edellytettiin, että koetut ongelmat eivät saa esiintyä CA Verify Automated Regression Testing for CICS- työkalun kanssa, jos se aiotaan hankkia yritykseen. Yrityksessä sovellusta päätettiin käyttää pääosin regressiotestauksen automatisointiin, vaikka työkalu sisältääkin mahdollisuudet myös muunlaisen testaamisen helpottamiseen. Työkalun koekäyttöä varten varattiin Yrityksen X resursseista yksi systeemitestiympäristö johon tuote asennettiin. Asennuksen suoritti CA yhdessä Yrityksen X asiakkaan kanssa, koska tuote asennettiin asiakkaan järjestelmään ja lopulta tuotteen hankinnasta päätetään yhdessä asiakkaan kanssa. Alustavasti koekäyttölisenssi myönnettiin 30 päiväksi. Pääasiallisena koekäyttäjänä toimi kirjoittaja.

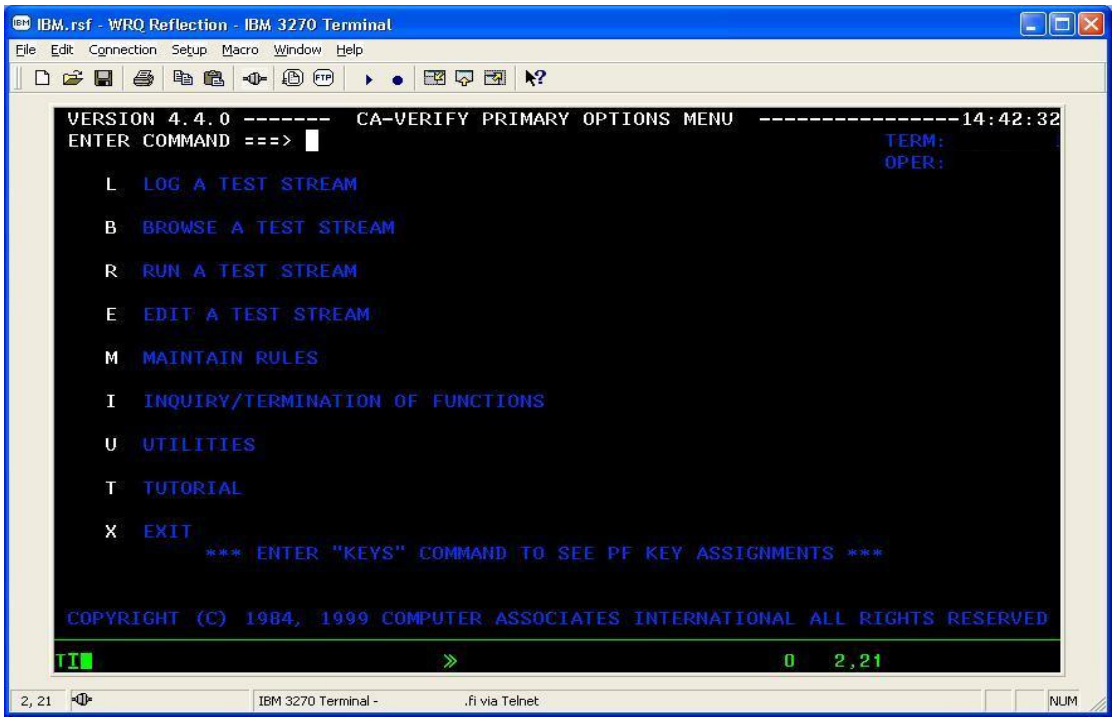
## 6.3 Sovellus Verify for CICS

CA Verify Automated Regression Testing for CICS on 3270-tyyppisten Online- sovellusten testausväline. Sitä voidaan käyttää testi- ja tuotantoympäristöissä ja se on pääasiallisesti tarkoitettu sovelluskehittäjien ja testaajien apuvälineeksi. Väline nauhoittaa sovelluksen käyttöä yhdeltä tai useammalta käyttäjältä tai päätteeltä, josta saatu nauhoitus voidaan ajaa muutettua sovellusta vasten. Ajettaessa CA Verify Automated Regression Testing for CICS ilmoit-

taa mahdollisista poikkeamista, jolloin testaaja voi päättää miten poikkeaman kanssa toimitaan. Sovellukseen tehtävät muutokset voidaan määritellä etukäteen sääntöjen avulla, jolloin huomioidaan vain odottamattomat muutokset. Verify Automated Regression Testing for CICS on siis yksinkertaisesti ilmaistuna nauhuri. Käyttäjä käynnistää nauhurin, suorittaa haluamansa testit ja lopettaa nauhoituksen halutessaan. Näin syntynyttä testidataa päästään nauhoituksen jälkeen tarkastelemaan, muokkaamaan ja ajamaan uudestaan. (CA edustaja haastattelu, 2008.)

CA Verify Automated Regression Testing for CICS auttaa automatisoimaan ja suorittamaan erilaisia testejä IBM CICS –transaktiopalvelimen z/OS- sovelluksille jotka käyttävät 3270-tyyppisiä terminaaleja tai pääte-emulointia. Sovelluksella pystytään automatisoimaan regressio-, yksikkö-, stressi-, yhtäaikaisuus-, migraatio- ja systeemitestausta. Käyttämällä Verify for CICS –sovellusta voidaan varmistaa että CICS –palvelimen alla ajetut ohjelmat toimivat oikein sekä testi- että tuotantoympäristössä. Lisäksi automaatio-ominaisuuksilla voidaan alentaa kuluja laadunvarmistuksessa ja välttää tuotantovirheitä sekä järjestelmän downtimeä. (CA Verify Automated Regression Testing for CICS- tuoteseloste, 2008.)

CA:n edustaja korostaa, että Yrityksessä X käytettävän järjestelmän näytöt saattavat sisältää useita rivejä tärkeää tietoa, ja virheet saattavat olla vaikeita huomata ihmissilmällä näytöltä. Kun automaattinen sovellus vertaa testituloksia, se huomaa pienimmätkin virheet näytöillä. (CA edustaja haastattelu, 2008.)




Kuva 4. Verify for CICS päävalikko.

Päävalikon toiminnot ohjaavat nauhoitukseen liittyviä ominaisuuksia. ”Log a test stream”- toiminnolla voidaan nauhoittaa testidataa yhden tai useamman käyttäjän päätteeltä. ”Browse a test stream”- toiminto antaa mahdollisuuden selata ja tarkastella jo nauhoitettua testidataa. Jos nauhoitettu testi halutaan toistaa esimerkiksi uudella versiolla sovelluksesta, käytetään ”Run a test stream”- toimintoa. ”Edit a test stream”- toiminto mahdollistaa nauhoitetun testidatan muokkaamisen esimerkiksi turhien näytönkuvien poistamista varten. Testien nauhoituksessa ja toistossa käytettäviä sääntöjä voidaan hallinnoida ”Maintain rules”- toiminteen alla. Säännöillä voidaan ohittaa näyttöjen vertailussa esimerkiksi päivämääräkentät. Sääntöjä ei ole pakko käyttää, jolloin sovellus pysähtyy jokaiseen poikkeukseen alkuperäisen nauhoitetun näytön ja toistossa esiintyneen näytön välillä. Näin testaaja voi määritellä mikä poikkeus on oikea ja mikä väärä. ”Inquiry/Termination of functions”- valikon alta voidaan hallita käynnissä olevia toistoja. ”Utilities”- valikosta voidaan muokata testejä eri tavoilla (Copy, Rename, Delete, Update) ja yhdistää jo nauhoitettuja testejä yhdeksi jonoksi lisäämällä testejä toistensa perään. Tällöin voidaan ajaa monta testitapausta yhdellä ajokerralla.

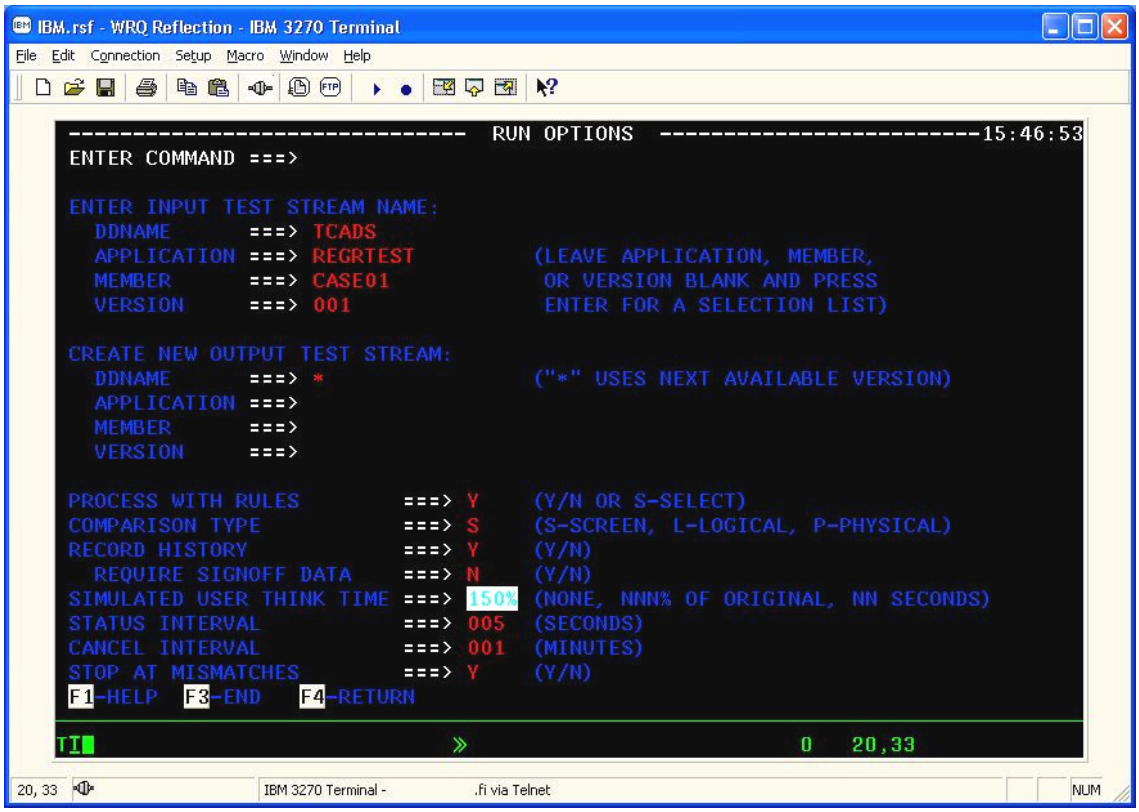
## 6.4 Koekäyttö

CA Automated Regression Testing for CICS- tuotteen koekäytössä pyrittiin selvittämään, miten väline soveltuu Yrityksen X regressiotestaukseen. Tarkkailtavia asioita oli tuotteen kattavuus, luotettavuus, kapasiteetti, opittavuus, käytettävyys, suorituskyky ja yhteensopivuus. Koekäytön alussa huomattiin muutamia käyttöoikeuspuutteita. Sovelluksen joihinkin toimintoihin on määritettävä käyttöoikeudet erikseen ja CA Verify Automated Regression Testing for CICS tarvitsee oman käyttäjätunnuksen Yrityksen X järjestelmään toimiakseen oikein. Sovellukselle luotiin käyttäjätunnus ja muut käyttöoikeusasiat hoidettiin asiakkaan edustajan kanssa kuntoon. Koekäytön aikana paikalla oli osan ajasta CA:n edustaja, joka piti myös demonstraation sovelluksen käyttämisestä Yrityksen X testausryhmälle. CA:n edustaja auttoi käyttöönotossa ilmenneissä ongelmissa koekäytön aikana. Koekäyttö suoritettiin Yrityksen X toimitiloissa.

Koekäyttö aloitettiin nauhoittamalla muutamia satunnaisia testitapauksia Verify- sovellukseen. Kun tapauksia oli nauhoitettu, kokeiltiin nauhoitettujen tapauksien ajamista alustavasti. Tämän tarkoitus oli tehdä ensimmäinen havainto sovelluksen toiminnasta ja yhteensopivuudesta käytettävään järjestelmään.

Seuraavaksi regressiotestipaketista valittiin kuusi testitapausta, joissa pyrittiin mahdollisimman monipuoliseen kattavuuteen. Tapaukset nauhoitettiin sovellukseen ja tarkasteltiin miten ne toistetaan CA:n työkalulla. Tapauksia toistettaessa huomattiin ensin ongelmia. Nauhoitettu näyttö ei vastannut tapauksen toistossa tullutta näyttöä. Ongelmasta raportoitiin CA:n edustajalle, joka saapui paikalle tarkastelemaan ongelmaa. Ongelman todettiin johtuvan järjestelmän vastausviiveestä. Verify Automated Regression Testing for CICS teki nauhoitetut toi-

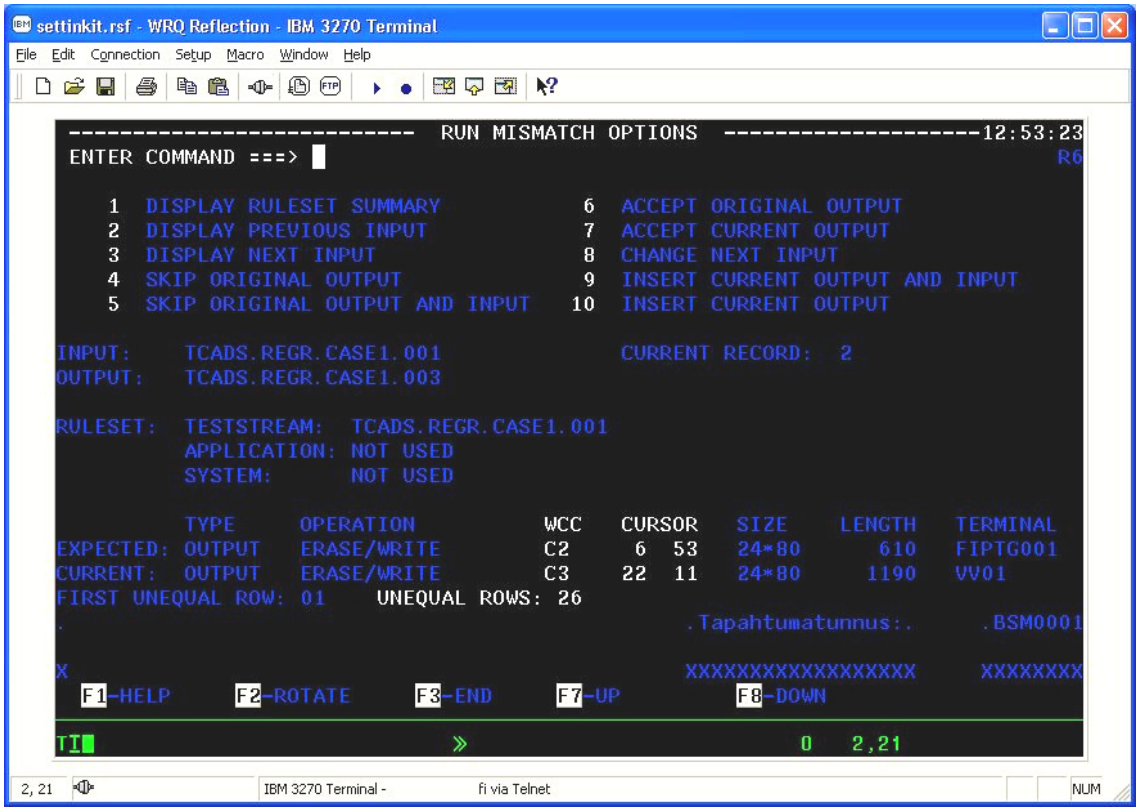
minnot millisekuntien viiveellä, ennen siirtymistä seuraavaan toimintoon, kun varsinaisessa manuaalisessa nauhoituksessa toiminnon suorittamiseen saattoi kulua esimerkiksi viisi sekuntia. Sovelluksessa voidaan kuitenkin simuloida käyttäjän vasteaikaa testitoiminnoissa, joka asetettiin testitapauksen toistamisen ajaksi 150%:iin alkuperäisestä. Tällä pystyttiin ohittamaan kyseinen ongelma ja testejä voitiin ajaa normaalisti ilman ongelmia. Vasteajan sopiva määrä todettiin kokeilemalla useita vaihtoehtoja. Seuraavassa kuvassa asetetaan vasteaika ja Status interval, jolla saadaan toiston aikana raportti toiston etenemisestä viiden sekunnin välein.




Kuva 5. Run options –näyttö, vasteajan asetus.

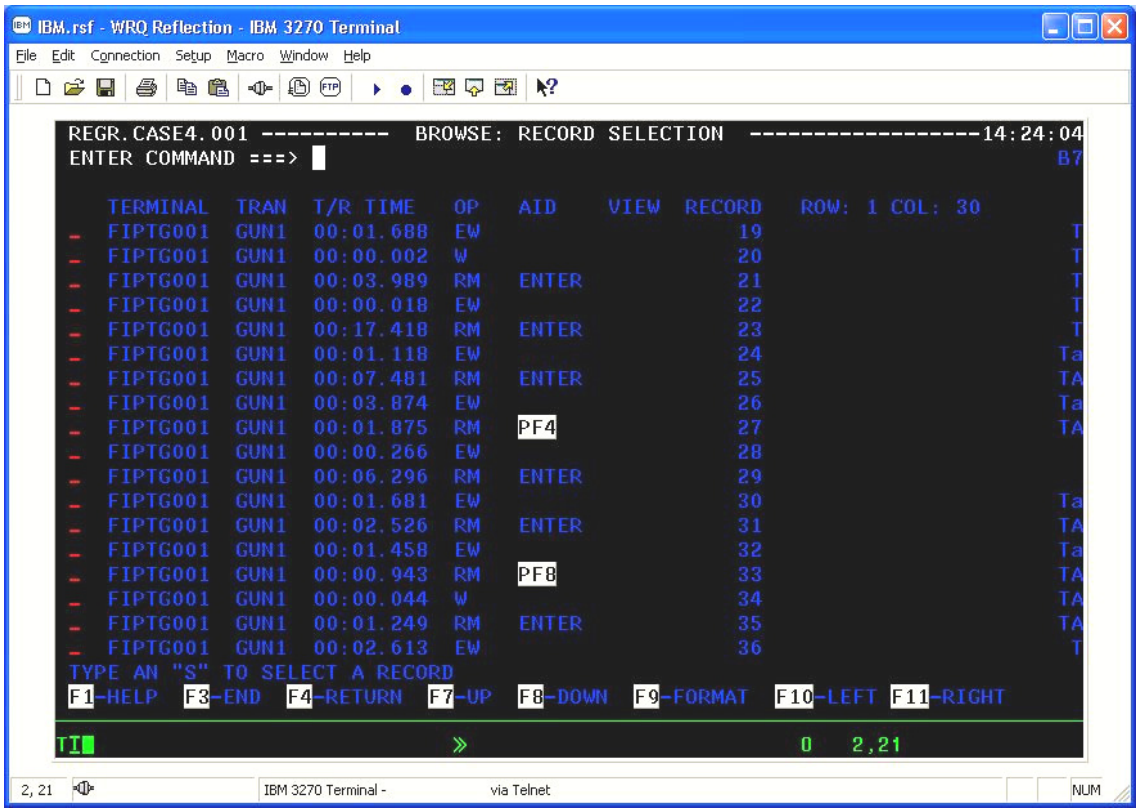
Tapauksien toiston todettiin luovan systeemitestiympäristön tietokantaan identtisen tilanteen, kuin mitä käsin tehdyissä, nauhoitetuissa tapauksissa luotiin. Ensin tilanteita ajettiin yksitellen ja sen jälkeen niitä järjesteltiin peräkkäin yhdeksi isoksi testitapaukseksi ”Utilities”- valikon kautta. Myös tällöin tapauksen toisto jäljitteli manuaalisen, nauhoitetun testin tulosta. Joidenkin testitapausten kohdalla todettiin, että CA Verify Automated Regression Testing for CICS ei pystynyt tekemään tarvittavia alkuvalmisteluita testejä varten. Näin manuaaliselta työltä ei täysin vältytty, joskin se oli luonteeltaan lähinnä nopeaa alkuvalmistelua.

Jos toistossa esiintyvissä näytöissä havaitaan eroavaisuuksia, CA Verify Automated Regression Testing for CICS pysähtyy ja tuo eroavaisuuden käyttäjän tarkasteltavaksi. Käyttäjä voi tällaisessa tilanteessa päättää, onko eroavaisuus sallittu muuttuja vai onko kyseessä oikea virhe.




Kuva 6. Sovellus on huomannut virheen ja tuo sen käyttäjän tietoon. Eroava kohta on merkitty X-kirjaimilla.

Kun testit oli nauhoitettu ja toistettu onnistuneesti, tarkkailtiin testidataa ”Browse”- toiminnon avulla. Selailussa voi tarkastella nauhoitettuja näyttöjä ja käyttäjän toimintaa näytöillä. Käyttäjän jokainen napinpainallus tallennetaan testidataan. Nauhoitettujen näyttöjen ja toistossa esiintyneiden näyttöjen välillä esiintyi ristiriitaa. CA:n työkalu yhdisteli kahdesta eri näytöstä osia selaustoimintoon. Asiasta raportoitiin CA:n edustajalle, joka ryhtyi selvittämään asiaa. Myöhemmin todettiin että virhe oli lähinnä kosmeettinen ja että sovellus teki kuitenkin näyttöjen vertailun oikein. Seuraavassa kuvassa selausnäytöllä on 36 tietuetta jossa jokaisella tietueella on tehty jokin toiminto järjestelmässä. Eroavaisuus huomattiin kun tarkasteltiin tietueita erikseen. Sovellus nauhoitti tietueita myös esimerkiksi näytöllä liikuttaessa, joka on käytännössä tarpeetonta dataa, mutta vaaditaan jotta sovellus osaa toistaa nauhoitetut testit mahdollisimman tarkasti.




Kuva 7. Testidatan selaus Browse- toiminnolla. Näytöllä on tietoja käyttäjän toiminnoista ja järjestelmään tehtävistä toiminnoista. Jokaista tietuetta voidaan tarkastella erikseen.

TRAN- kenttä ilmaisee järjestelmässä käytetyn transaktion tyypin.

T/R TIME kertoo järjestelmän vastausviiveen.

OP kertoo suoritetun operaation tyypin:

**W:** write

**EW:** erase/write

**EWA:** erase/write alternate

**RB:** read buffer

**RM:** read modified

**RMA:** read modified all

**EAU:** erase all unprotected

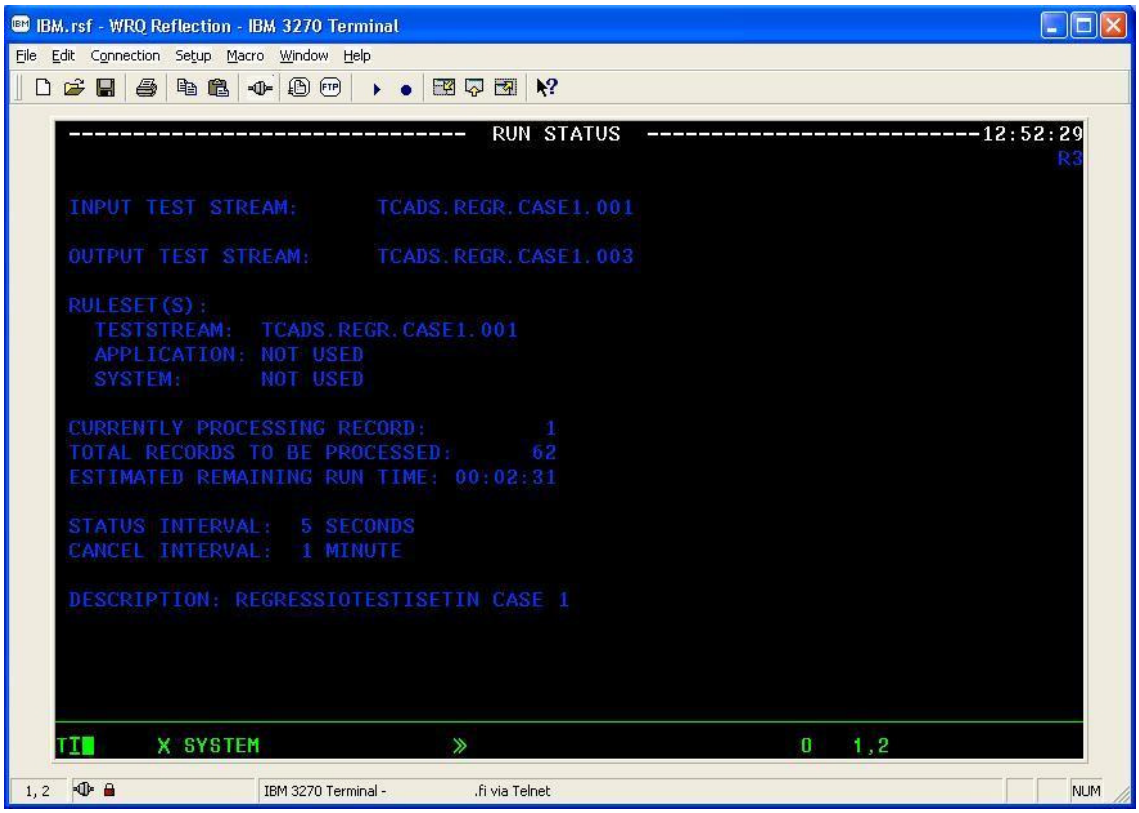
**WSF:** write structured field

**CPY:** copy

**RD:** read

AID- kenttä näyttää käyttäjän painaman näppäimen jokaisen toiminnon kohdalla.

Aikaa testiympäristön valmisteluun, aloituspalavereihin ja työkalun käytön opetteluun kului noin kaksi viikkoa. Regressiotestitapauksen suorittamiseen kuluu manuaalisesti keskimäärin puoli tuntia, tapauksesta riippuen. Sama aika vaaditaan siis myös testin nauhoittamiseen työkaluun. Testejä on kuitenkin useassa tapauksessa tarpeellista nauhoittaa vain kerran, koska nauhoitettua testidataa voi muokata tarvittaessa. Näin myös ylläpidossa säästetään aikaa. CA:n työkalulla testien ajamiseen kulunut aika on huomattavasti pienempi kuin manuaalisesti; kuuden testitapauksen ajamiseen työkalulla kului yhteensä hieman yli 10 minuuttia. Manuaalisesti testaajalta olisi arviolta kulunut aikaa kolme tuntia. Ajan säästössä pitää kuitenkin ottaa huomioon mahdolliset manuaaliset alkuvalmistelut, joihin kuudessa testitapauksissa kului aikaa yhteensä noin kymmenen minuuttia. Näin kokonaisajaksi saadaan 20 minuuttia, joka on noin 11 prosenttia manuaaliselle suorittamiselle lasketusta ajasta.




Kuva 8. Toiston aikana sovellus näyttää tietoja prosessin etenemisestä.

Koska testauksen perimmäinen tarkoitus on löytää virheitä, työkalun koekäytössä luotiin tahallisia virhetilanteita systeemitestin tietokantaan ja tarkkailtiin löytääkö sovellus virheet. Koekäytön kesto oli rajoitettu lisenssin kestoon, mutta aika riitti alustavasti työkalun ominaisuuksien ja yhteensopivuuden toteamiseen. Koekäytön lopuksi kirjoittaja laati CA Verify Automated Regression Testing for CICS- työkalun käytöstä käyttöohjeen Yrityksen X regressiotes-

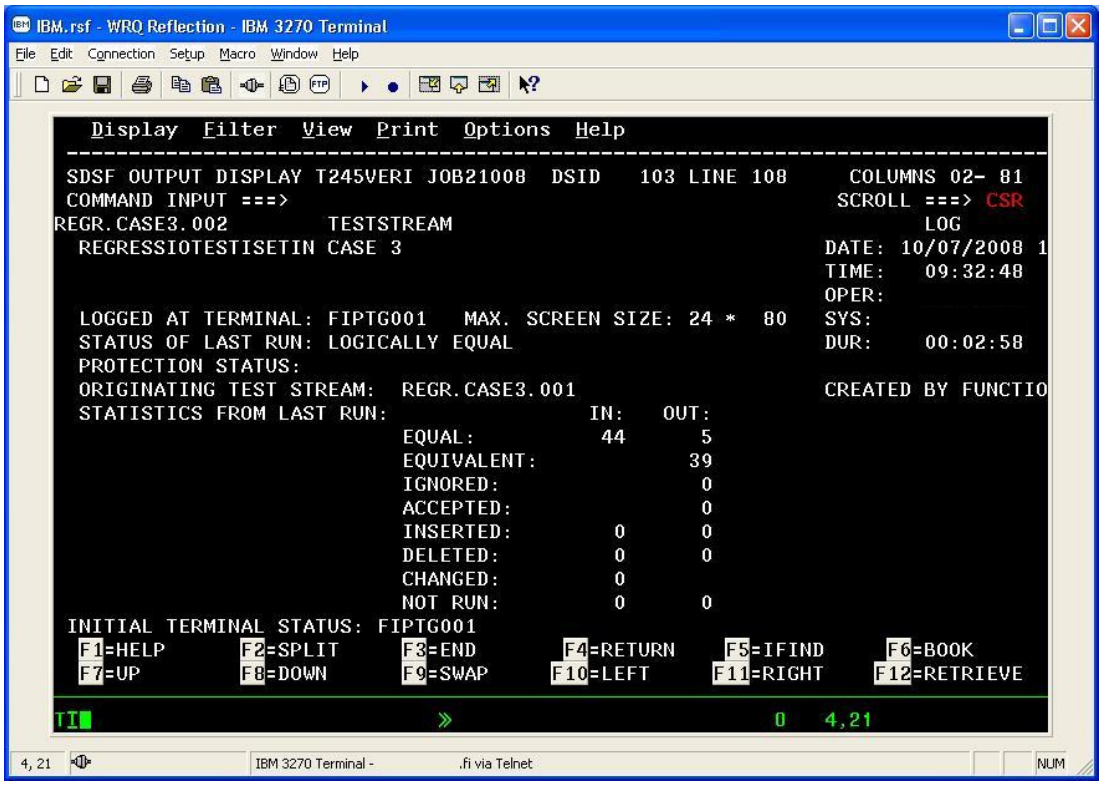
taukseen. Ohjeen tarkoituksena oli tutustuttaa mahdollisia käyttäjiä etukäteen sovellukseen ja toimia osittain dokumentaationa koekäytöstä. Käyttöohje on tutkimuksen liitteenä.

Koekäytön yhteydessä luotiin CA Verify Automated Regression Testing for CICS- tuotteen manuaalin perusteella JCL-eräajo, jonka ajaminen tuottaa raportin. Raportista nähdään tilastoja ajettujen testien lopputuloksista. Lisäksi raportista voidaan tarkastella jokaista nauhoitettua näyttöä ja verrata sitä odotettuun näyttöön sekä alkuperäiseen nauhoitettuun näyttöön. JCL eli Job Control Language määrittelee, millä tavalla ohjelmat suoritetaan Mainframe- koneella. JCL- toiminnot ovat rajapinta ohjelmien ja käyttöjärjestelmän välillä.

```
000001 //T245VERI  JOB (249,7D),LAX,CLASS=C,
000002 //      NOTIFY=ABC123,MSGCLASS=1,MSGLEVEL=(1,1)
000003 //*--------------------------------------------------------------------*
000004 //*                                                                    *
000005 //*       VERIFY RAPORTTI
000006 //*       MUUTA  RIVILLE 19 TESTISTREAMISI NIMI
000007 //*       SELECT APPL.MEMBER.00X
000008 //*       PRINT SÄÄNNÖILLÄ MÄÄRITELLÄÄN OUTPUTIN ULKOASU, LUE MANUAALI
000009 //*
000010 //*--------------------------------------------------------------------*
000011 //TCABATCH  EXEC PGM=TCABATCH,REGION=2M
000012 //STEPLIB   DD DSN=CICS.VERIFYC.LOAD31,DISP=SHR
000013 //TCADSIN   DD DSN=CICS.VERIFYC.TCADS,DISP=SHR
000014 //TCADSOUT  DD DSN=CICS.VERIFYC.TCADS,DISP=SHR
000015 //SYSPRINT  DD SYSOUT=1
000016 //TCAPRINT  DD SYSOUT=1
000017 //SYSIN     DD *
000018 PRINT APRULES(YES) DIFF(YES)
000019 SELECT REGR.CASE3.002
```

Kuva 9. JCL- eräajo jolla luodaan raportti.

Seuraavassa kuvassa on JCL-eräajolla tuotetun raportin etusivu. Raporttia voidaan selailla F7 ja F8 toimintonäppäimillä. Raportti sisältää tilastoja mm. seuraavista asioista: ajettujen testien määrä, todellisuudessa ajettujen testien määrä, läpimenneiden testien määrä, virheeseen päätyneiden testien määrä, kesken jääneiden testien määrä, selitykset kesken jääneiden testien virheistä, käytetyt resurssit (koneteho jne.) ja käytetty aika.




Kuva 10. Eräajon tuottama raportti.

## 6.5 Koekäytön tuloksia

Talc2000- työkalun kanssa ilmenneitä ongelmia (ks. kohta 7.1) tarkkailtiin erityisesti myös CA Verify Automated Regression Testing for CICS:in koekäytössä. CA:n työkalun toiminnassa päätе-emulaattorin kautta ei huomattu ongelmia. Sovellukseen pääsee miltä tahansa työasemalta jossa on Reflection- pääte-emulaattori ja käyttäjällä on tarvittavat tunnukset. Talc2000:n kanssa koettuja luotettavuusongelmia ei myöskään esiintynyt CA:n työkalun kanssa. Samat testit toistettiin useita kertoja samaan tietokantatilanteeseen ja tulokset olivat aina samat. Ympäristöön luotiin tarkoituksellisesti virhetilanteita ja tarkistettiin että sovellus huomaa virheet. Talc2000- työkalun käytön yhteydessä huomattiin että ylläpito oli raskasta, koska jos järjestelmään tehtiin muutoksia, tuli testeihin käytettävät testitapaukset nauhoittaa uudelleen. CA Verify Automated Regression Testing for CICS sisältää mahdollisuuden syöttää sääntöjä joita vasten testit ajetaan. Jos järjestelmään tehdään jokin tietty muutos, se voidaan määrittää jälkikäteen jo nauhoitetulle testitapaukselle kun sitä toistetaan uudestaan. Näin testitapauksia ei välttämättä tarvitse nauhoittaa uudelleen ja säästetään suuri määrä työtä jota Talc2000:n kanssa tehtiin. Työkalu koettiin toimivaksi ja helppokäyttöiseksi. Koekäytön jälkeen hankinnan riskeiksi eroteltiin resurssiasiat, todellinen ajansäästö ja kustannusasiat. Työkalu vaatisi hankittaessa vastuuhenkilön sekä tälle varahenkilön, jotka pitäisivät yllä nauhoituksia ja huolehtisivat regressiotestauksen toteutuksesta. Todellista ajansäästöä on vaikea

arvioida, jos työkalu hankittaisiin, koska koekäyttö raapaisi käytännössä vain pintaa ja totesi työkalun toimivuuden. Ajansäästöstä tehtiin alustavia havaintoja osalla regressiotestitapauksista, ja havainnot olivat positiivisia. Jos ajansäästö olisi samaa luokkaa koko regressiotestipaketilla, olisi työkalun hankinta jo kustannussyistä järkevää. Jos työkaluun päätettäisiin sijoittaa, siitä tulisi olla todellista näkyvää hyötyä pitkälläkin tähtäimellä. Kohdejärjestelmä alkaa olla elinkaarensa päässä ja jos se tullaan korvaamaan lähivuosina, on hankinnan järkevyyttä syytä pohtia uudestaan.

# 7 Lopputulokset ja pohdintaa

Yrityksen X regressiotestausprosessissa huomattiin puutteita sekä kyselyssä että kirjallisuuteen perustuvissa havainnoissa. Aiheeseen liittyvä kirjallisuus korostaa regressiotestauksen tärkeyttä etenkin monimutkaisissa järjestelmissä, jollainen yrityksellä on käytössä. Myös regressiotestauksen automatisointia suositellaan laajasti lähdekirjallisuudessa. Regressiotestausta helpottamaan koekäyttöön otettu CA Verify Automated Regression Testing for CICS antoi hyvän vaikutelman ja näytti potentiaalinsa, vaikka koekäytössä ei edes ehditty käyttämään kaikkia sen ominaisuuksia. CA:n työkalu todettiin koekäytön perusteella luotettavaksi ja yhteensopivaksi yrityksen järjestelmään. Myös helppokäyttöisyys tuli koekäytössä esille. Koekäytön kesto oli noin 30 päivää. Tämä aiheutti joitain käytännön rajoituksia työkalun ominaisuuksien selvittämiselle, mutta alustavat havainnot ehdittiin tehdä. Koekäytön aikana kirjoittaja oppi käyttämään työkalua sujuvasti. Suurin kysymys työkalun kanssa on tarvittavan ylläpidon määrä ja siihen liittyvät resurssiasiat. Yrityksen X resursseista tulisi nimittää työkalua ja regressiotestauksen nauhoituksia hoitava henkilö sekä hänelle varahenkilö. Yrityksen X testaajia resursoidaan ensisijaisesti projekteihin, mikä voi vaikeuttaa sopivan vastuuhenkilön nimittämistä. Lisäksi tässä tapauksessa on otettava huomioon kohdejärjestelmän elinkaari. Yrityksessä käytettävä järjestelmä on perustettu 1980-luvulla ja sitä on yritetty korvata uudella järjestelmällä tuloksetta. Jos järjestelmä tullaan korvaamaan muutaman vuoden sisällä, ei työkalun hankinta ole kustannussyistä järkevää. Jos väline hankittaisiin, sen käyttöä voitaisiin laajentaa esimerkiksi ohjelma- tai suorituskykytesteihin. Joka tapauksessa regressiotestaukseen tulisi kiinnittää enemmän huomiota. Jos välinettä ei hankita, tulisi regressiotestausprosessia yrittää parantaa muilla keinoilla. Regressiotestaukseen käytettäviä testitapauksia ei ole vielä siirretty Quality Centeriin. Tämä tulisi tehdä, jotta regressiotestauksen seuranta helpottuisi ja päästäisiin samaan käytäntöön, jota muissa testeissä käytetään. Testitapausten saamiseksi Quality Centeriin on esimiehelle tehty ehdotus Regressiotestaus -aliprojektin perustamisesta hakemistorakenteeseen. Lisäksi testitapaukset tulisi päivittää ajan tasalle — jotkin tapaukset eivät käyttäydy odotetusti, koska tapausten luonnin jälkeen järjestelmään on tullut useita muutoksia. Suurin ongelma lienee kuitenkin resurssien ja ajan puute.

Lähteet

Black, R. 2004. Critical testing processes. Boston: Addison-Wesley.

CA Verify Automated Regression Testing for CICS product brief. CA Inc. Tulostettu 10.12.2008.
http://ca.com/files/ProductBriefs/verify_for_cics_product_brief_us.pdf

CA edustaja haastattelu. Joulukuu 2008 & Tammikuu 2009. Espoo.

Dustin, E. 2003. Effective software testing – 50 specific ways to improve your testing. Kuudes painos. Boston: Pearson Education Inc.

Farrell-Vinay, P. 2008. Manage software testing. New York: Auerbach Publications.

Fewster, M. & Graham, D. 1999. Software test automation – Effective use of test execution tools. Great Britain: ACM press.

Gao, J. 2002. Software regression testing. Luettu 1.9.2009.
www.engr.sjsu.edu/gaojerry/course/287/**regression-test**.ppt

Grove Consultants. 2001. ISEB Foundation Certificate in Software Testing v.1.5a. (koulutusmateriaali).

Haikala, I. & Märijärvi, J. 2004. Ohjelmistotuotanto. Helsinki: Talentum.

Hinkkanen, J. & Leiman, M. 2008. Testing Forum 08 Seminaarimateriaali 1-2. Helsinki.

Hirsjärvi, S., Remes, P. & Sajavaara, P. 2007. Tutki ja kirjoita. 13., osin uudistettu painos. Helsinki: Tammi.

IBM. 2004. CICS – an introduction. Luettu 11.12.2008.
ftp://service.boulder.ibm.com/software/htp/cics/PDF/cics_introduction.pdf

IEEE Standardi 610.12 1990. IEEE Standard Glossary of Software Engineering Terminology.

Pettichord, B. 2001 (a). Seven Steps to Test Automation Success. Luettu 10.12.2008.
http://www.io.com/~wazmo/papers/seven_steps.html

Pettichord, B. 2001 (b). Success with Test Automation. Luettu 10.12.2008.
http://www.io.com/~wazmo/succpap.htm

Pyhäjärvi, M. 2006. Katsaus testausautomaation toteutuksiin. Luettu 12.12.2008.
http://www.tol.oulu.fi/projects/wg4/minutes/WG4040506_MaaretPyhajarvi.pdf

Pyhäjärvi, M. & Pöyhönen, E., 2004. Testausvälineet ja testauksen automatisointi. Tulostettu 24.11.2008.
http://users.jyu.fi/~sakkinen/testaus/aineisto/8_TestausvalineetJaTestauksenAutomatisointi_v0_1.ppt

Tamres, L., 2002. Introducing Software Testing. Harlow: Pearson Education Ltd.

Kuvaluettelo

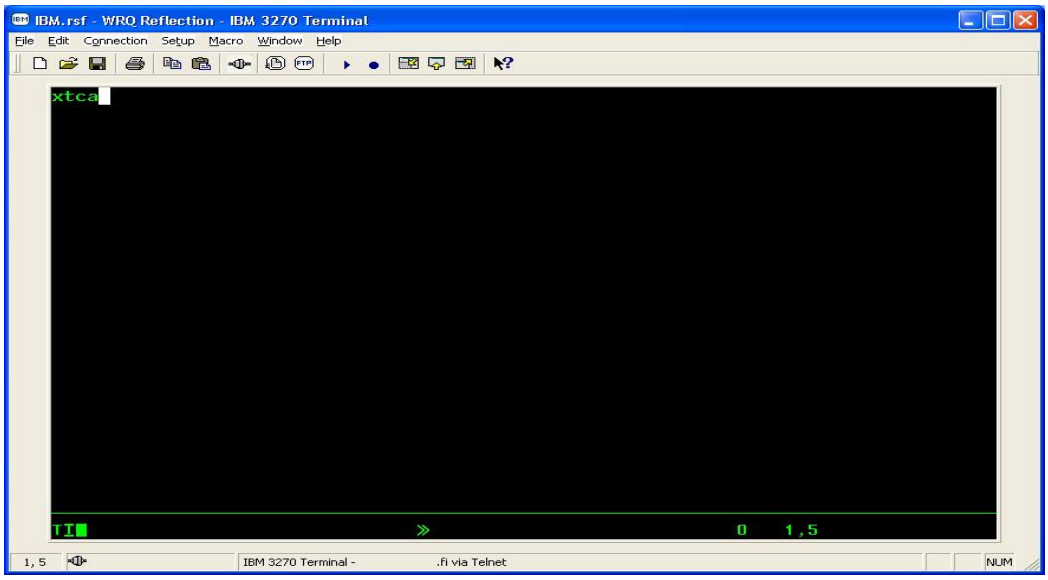


Liitteet

**OHJE**
VERIFY AUTOMATED REGRESSION TESTING FOR CICS

Palauta systeemitestiympäristöön haluamasi backup (perusversio), muuta datesimulaattorilla sopiva päivämäärä testejä varten.

Kirjaudu järjestelmän tyhjällä näytöllä CA Verify -sovellukseen transaktiolla XTCA.




<u>Nauhoitus ja selaaminen</u>

Jos haluat nauhoittaa testiä, valitse päävalikossa **L** (Log a test stream)

```
ENTER COMMAND ===>                                          TERM: FIPTG002
                                                            OPER: U002BAF
   L  LOG A TEST STREAM

   B  BROWSE A TEST STREAM

   R  RUN A TEST STREAM

   E  EDIT A TEST STREAM

   M  MAINTAIN RULES

   I  INQUIRY/TERMINATION OF FUNCTIONS

   U  UTILITIES

   T  TUTORIAL

   X  EXIT
```

Valitse mistä haluat nauhoittaa, yleensä T eli This Terminal. Muilla vaihtoehdoilla voit nauhoittaa muitten tekemiä toimintoja.

```
ENTER COMMAND ===>

   T  THIS TERMINAL

   O  ANOTHER TERMINAL

   M  MULTIPLE TERMINALS
```

Syötä application- kenttään esimerkiksi oma nimesi ja member kenttään haluamasi määrite, esimerkiksi Case01. Syötä description kenttään kuvaus nauhoituksesta, maksimissaan kolme riviä. Älä valitse Stop optioniksi PF-näppäintä, koska teet todennäköisesti testissä toimintoja näillä näppäimillä. Jätä valinta manuaaliseksi (MAN). Muut valinnat voi jättää oletukseksi (N).

```
LOG TEST STREAM AS:
  DDNAME      ===> TCADS
  APPLICATION ===> REGR
  MEMBER      ===> CASE1
  VERSION     ===> 002

DESCRIPTION ===> Regressiotestisetin case 1
            ===> Palvelun muutos
            ===>


STOP OPTION              ===> MAN         (MAN, PF__, PA_, OR CLEAR)
PROCESS WITH RULES       ===> N           (Y/N)
TEST STREAM PROTECTION   ===>             (R-READ W-WRITE P-PRINT)
LOG INPUT SCREENS ONLY   ===> N           (Y/N)
EXTEND TEST STREAM       ===> N           (Y/N)

RULESET NAME:
```

Paina enter ja nauhoitus alkaa järjestelmän tyhjältä näytöltä. Jatka normaalisti järjestelmän transaktiolla. Tee haluamasi testit ja kun haluat lopettaa, palaa järjestelmän tyhjälle näytölle ja kirjoita **xtca stop.**

Voit mennä tarkastelemaan nauhoittamaasi streamia **B** (Browse) komennolla päävalikossa. Valitse haluamasi testijono ja paina enteriä päästäksesi eteenpäin selauksessa. Kolmas enter tuo näytölle listan nauhoitetuista näytöistä. Voit tarkastella näyttöjä toiminnolla S. Nauhoitetuilla näytöillä pääsee liikkumaan eteen ja taakse näppäimillä PF5/PF6.

```
REGR.CASE1.001 ----------  BROWSE: RECORD SELECTION  ------------------14:05:21
ENTER COMMAND ===>                                                      B7

   TERMINAL  TRAN  T/R TIME   OP   AID    VIEW  RECORD   ROW: 1 COL: 30
_  FIPTG001  GUN1  00:02.618  RM   ENTER             1
_  FIPTG001  GUN1  00:00.004  EW                     2                      T
_  FIPTG001  GUN1  00:04.062  RM   ENTER             3                      T
_  FIPTG001  GUN1  00:00.195  EW                     4                      T
_  FIPTG001  GUN1  00:00.001  W                      5                      T
_  FIPTG001  GUN1  00:01.686  RM   ENTER             6                      T
_  FIPTG001  GUN1  00:00.014  EW                     7                      T
_  FIPTG001  GUN1  00:21.218  RM   ENTER             8                      T
_  FIPTG001  NMP1  00:00.249  EW                     9
_  FIPTG001  NMP1  00:00.001  W                     10
_  FIPTG001  NMP1  00:02.013  RM   PF19             11
_  FIPTG001  NMP1  00:00.068  EW                    12
_  FIPTG001  NMP1  00:00.001  W                     13
_  FIPTG001  NMP1  00:03.602  RM   PF9              14
_  FIPTG001  GUN1  00:00.807  EW                    15                      T
_  FIPTG001  GUN1  00:00.003  W                     16                      T
_  FIPTG001  GUN1  00:00.341  RM   ENTER            17                      T
_  FIPTG001  GUN1  00:00.056  EW                    18                      T


               TYPE AN "S" TO SELECT A RECORD
```

Tran –kenttä näyttää millä transaktiolla kussakin recordissa on liikuttu.

T/R time kertoo käyttäjän mietintäajan tai output screenissä järjestelmän vastausajan.

OP kertoo suoritetun operaation tyypin:

**W:** write

**EW:** erase/write

**EWA:** erase/write alternate

**RB:** read buffer

**RM:** read modified

**RMA:** read modified all

**EAU:** erase all unprotected

**WSF:** write structured field

**CPY:** copy

**RD:** read

Nauhoituksen ajaminen ja säännöt

Kun haluat ajaa nauhoituksia, valitse toiminto R (Run a test stream). Käytä toimintoa "*" (USES NEXT AVAILABLE VERSION) kohtaan DDNAME, jolloin Verify luo output- tiedostosta uuden version. Näin joka ajosta saadaan oma versionsa, joita voidaan vertailla myöhemmin.

**Tärkeää:** Valitse User think timeksi vähintään ”150%”. Jos valitset NONE, sovellus tekee jokaisen toiminnon millisekunneissa ja järjestelmä ei välttämättä ehdi vastaamaan. Tämä todettiin koekäytössä.

```
-------------------------------  RUN OPTIONS  -------------------------12:48:18
 ENTER COMMAND ===>

 ENTER INPUT TEST STREAM NAME:
   DDNAME      ===> TCADS
   APPLICATION ===> REGR             (LEAVE APPLICATION, MEMBER,
   MEMBER      ===> CASE1             OR VERSION BLANK AND PRESS
   VERSION     ===> 001               ENTER FOR A SELECTION LIST)

 CREATE NEW OUTPUT TEST STREAM:
   DDNAME      ===> *                ("*" USES NEXT AVAILABLE VERSION)
   APPLICATION ===>
   MEMBER      ===>
   VERSION     ===>

 PROCESS WITH RULES        ===> Y    (Y/N OR S-SELECT)
 COMPARISON TYPE           ===> S    (S-SCREEN, L-LOGICAL, P-PHYSICAL)
 RECORD HISTORY            ===> Y    (Y/N)
   REQUIRE SIGNOFF DATA    ===> N    (Y/N)
 SIMULATED USER THINK TIME ===> 150% (NONE, NNN% OF ORIGINAL, NN SECONDS)
 STATUS INTERVAL           ===> 005  (SECONDS)
 CANCEL INTERVAL           ===> 001  (MINUTES)
 STOP AT MISMATCHES        ===> Y    (Y/N)


               F1-HELP  F3-END   F4-RETURN
```

Voit valita valikosta käytätkö ajossa sääntöjä jotka olet luonut (suotavaa, koska muuten pysähtyy oletuksena jokaiseen muuttujaan kuten päivämäärään, käyttäjätunnukseen jne.) Voit luoda säännöt etukäteen päävalikon M (Maintain rules) komennolla, tai tehdä sääntöjä ajaessasi nauhoitusta. Verifyssä on mahdollisuus valita pysähtyykö sovellus jokaisen eroavaisuuden kohdalla. Jos valitset N, voit ajaa nauhoituksen läpi ja tarkastella näyttöjen eroja vasta lopussa. Jos valitset oletuksen Y, sovellus pysähtyy alkuperäisen tallennuksen ja nykyisen ajon erojen kohdalla ja antaa mahdollisuuden muokata sääntöjä lennosta ja jatkaa ajoa. Sääntöjä pääsee muokkaamaan toiminnolla 1 Display ruleset summary. PF2 Rotate toiminnolla pääset pyörittelemään näyttöjä, jolla verify on huomannut eroavaisuuden.

```
--------------------------  RUN MISMATCH OPTIONS  --------------------11:36:56
 ENTER COMMAND ===>                                                      R6

    1  DISPLAY RULESET SUMMARY           6  ACCEPT ORIGINAL OUTPUT
    2  DISPLAY PREVIOUS INPUT            7  ACCEPT CURRENT OUTPUT
    3  DISPLAY NEXT INPUT                8  CHANGE NEXT INPUT
    4  SKIP ORIGINAL OUTPUT              9  INSERT CURRENT OUTPUT AND INPUT
    5  SKIP ORIGINAL OUTPUT AND INPUT   10  INSERT CURRENT OUTPUT

INPUT:    TCADS.REGR.CASE2.001              CURRENT RECORD:  79
OUTPUT:   TCADS.REGR.CASE2.002

RULESET:  TESTSTREAM:  TCADS.REGR.CASE2.001
          APPLICATION: NOT USED
          SYSTEM:      NOT USED

          TYPE     OPERATION          WCC   CURSOR   SIZE    LENGTH   TERMINAL
EXPECTED: OUTPUT   WRITE              C3    21   2   24*80     1074   FIPTG001
CURRENT:  OUTPUT   WRITE              C3    21   2   24*80     1074   VV01
FIRST UNEQUAL ROW: 12    UNEQUAL ROWS: 02
.01     0811  Palvelu kytkentä 071008    6,474999    0,000000         L93071008
.01     0811  Palvelu kytkentä 071008    6,474999    0,000000         B68071008
                                                                      XXX


               F1-HELP     F2-ROTATE     F3-END     F7-UP         F8-DOWN
```

XXX merkkaa eroavaisuuden paikan näytöllä. Jos näytöllä on ------- jonkin kentän alla, se tarkoittaa että kenttä on merkitty säännöissä variableksi ja Verify tunnistaa sen. Muokataan ylläolevalle näytölle sääntö joka ohittaa kyseisen virheen. Näyttöön X:llä merkityt L93 ja B68 ovat käyttäjätunnuksia, joten ne voidaan määritellä muuttujiksi. Valitaan toiminto 1, display ruleset summary. Lisätään sääntöön RUL00001 määrite Insert (i)-komennolla, jotta Verify tunnistaa käsiteltävän näytön.

```
--------------------------  EDIT  RULES - SUMMARY ---------------------13:09:03
ENTER COMMAND ===>

RULE NAME: RUL00015           RULESET NAME:     TCADS.REGR.CASE1.001
LINE   1 TO  14 OF  41        TEST STREAM NAME: TCADS.REGR.CASE1.001

S-EDIT  I-INSERT  D-DELETE  R-REPLICATE  P-PREVIEW
  OBJECT      TYPE ROW  COL LEN OP  VALUE FROM THE MODEL SCREEN/DESCRIPTION
_ RULESET     T/S                   TCADS.REGR.CASE1.001
i  RULE                             RUL00001
```

Aukeaa valikko josta voidaan valita erilaisia keinoja näytön tunnistamiseksi ja kenttien määrittelemiseksi. Tunnistetaan näyttö Field recognition- komennolla ja määritetään muuttujat

Variable field- vaihtoehdolla. Tunnistus tehdään näytöllä viemällä kursori haluttuun kohtaan ja painamalla PF9 näppäintä.

```
---------------------- ADD RULES - RULE ACTIONS -----------------------13:11:00
ENTER COMMAND ===>

RULE NAME     ===> RUL00001   RULESET NAME:     TCADS.REGR.CASE1.001
                              TEST STREAM NAME: TCADS.REGR.CASE1.001

DESCRIPTION   ===>
              ===>

      1  FIELD RECOGNITION               11  GENERATE FIELD VALUE
      2  SCREEN RECOGNITION              12  INSERT SCREENS
      3  VARIABLE FIELD                  13  DELETE SCREENS
      4  DELETE FIELD                    14  CUT SCREEN FIELD
      5  MOVE FIELD                      15  PASTE SCREEN FIELD
      6  CHANGE FIELD VALUE              20  USERID LOGGING
      7  NEW FIELD                       21  TERMINAL ID LOGGING
      8  CHANGE AID KEY                  22  TRANSACTION ID LOGGING
      9  CHANGE CURSOR LOCATION
     10  CHANGE WCC VALUES
```

```
-------------------------  EDIT  RULES - SUMMARY ---------------------12:58:37
ENTER COMMAND ===>

RULE NAME: RUL00001            RULESET NAME:     TCADS.REGR.CASE1.001
LINE  38 TO  41 OF  41         TEST STREAM NAME: TCADS.REGR.CASE1.001

S-EDIT  I-INSERT  D-DELETE  R-REPLICATE  P-PREVIEW
  OBJECT      TYPE ROW  COL LEN OP  VALUE FROM THE MODEL SCREEN/DESCRIPTION
_   FLD-RECOG        3   31  10 EQ  PÄÄVALIKKO
_    VARIABLE OUT    3   60   4     .L93
_    VARIABLE OUT    3   64  11     .07.10.2008
_    VARIABLE OUT    3   75   6     .08:05
```

Säännössä näyttö on tunnistettu kentästä “Päävalikko” joka sijaitsee näytön rivillä 3 ja kolumnissa 31. Kentän pituus on 10 merkkiä. Tunnistuksen alle on lisätty muuttujia jotka ohitetaan kyseistä näyttöä prosessoidessa. Muuttujiksi on määritelty L93 käyttäjätunnus, 07.10.2008 päivämäärä ja 08:05 kellonaika. Kun sääntö on valmis, palataan PF3 näppäimellä jatkamaan ajoa. Jos ajo ei jatku automaattisesti, et ole lisännyt virheen ohittavaa sääntöä.

Jos samassa rulesetissä on päällekkäisiä sääntöjä, tämä ei haittaa, vaan Verify käyttää ensin vastaan tulevan ja jättää päällekkäisen huomioimatta.

## Utilities-toiminto

Utilities- valikosta voidaan muokata testejä eri tavoilla (Copy, Rename, Delete, Update) ja yhdistää jo nauhoitettuja testejä yhdeksi jonoksi lisäämällä testejä toistensa perään (Append records). Tällöin voidaan ajaa monta testitapausta yhdellä ajokerralla. Test stream- kenttään syötetään se testijono, jota halutaan muokata tai jonka perään halutaan lisätä testejä.

Utilities- valikon alla olevat toiminnot:

```
ENTER COMMAND ===>                                                    U1


   C  COPY A TEST STREAM                  A  APPEND RECORDS TO A TEST STREAM
   R  RENAME A TEST STREAM                I  INSERT RECORDS INTO A TEST STREAM
   D  DELETE A TEST STREAM                M  MERGE TERMINALS INTO A TEST STREAM
   U  UPDATE A TEST STREAM DIRECTORY

ENTER TEST STREAM NAME:
  DDNAME      ===> TCADS
  APPLICATION ===>                 (LEAVE APPLICATION, MEMBER,
  MEMBER      ===>                  OR VERSION BLANK AND PRESS


               VERSION     ===> 001              ENTER FOR A SELECTION LIST)
```

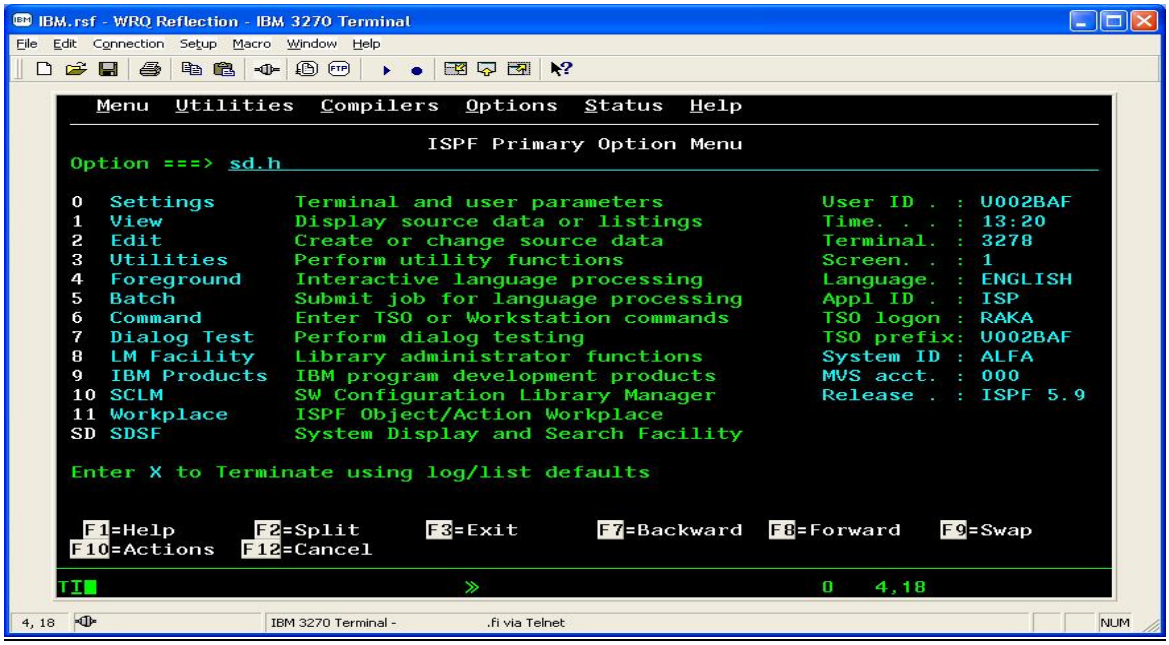
Siirrytty Append -komentoon:

```
REGR.CASEKAKS.002 ----------------  APPEND  ----------------------------10:31:33
ENTER COMMAND ===>                                                      UA

APPEND FROM TEST STREAM:
  DDNAME      ===> TCADS
  APPLICATION ===> REGR             (LEAVE APPLICATION, MEMBER,
  MEMBER      ===> CASE02            OR VERSION BLANK AND PRESS
  VERSION     ===> 002               ENTER FOR A SELECTION LIST)

OPTIONS:
 INCLUDE RULESET    ===> Y          (Y/N)
```

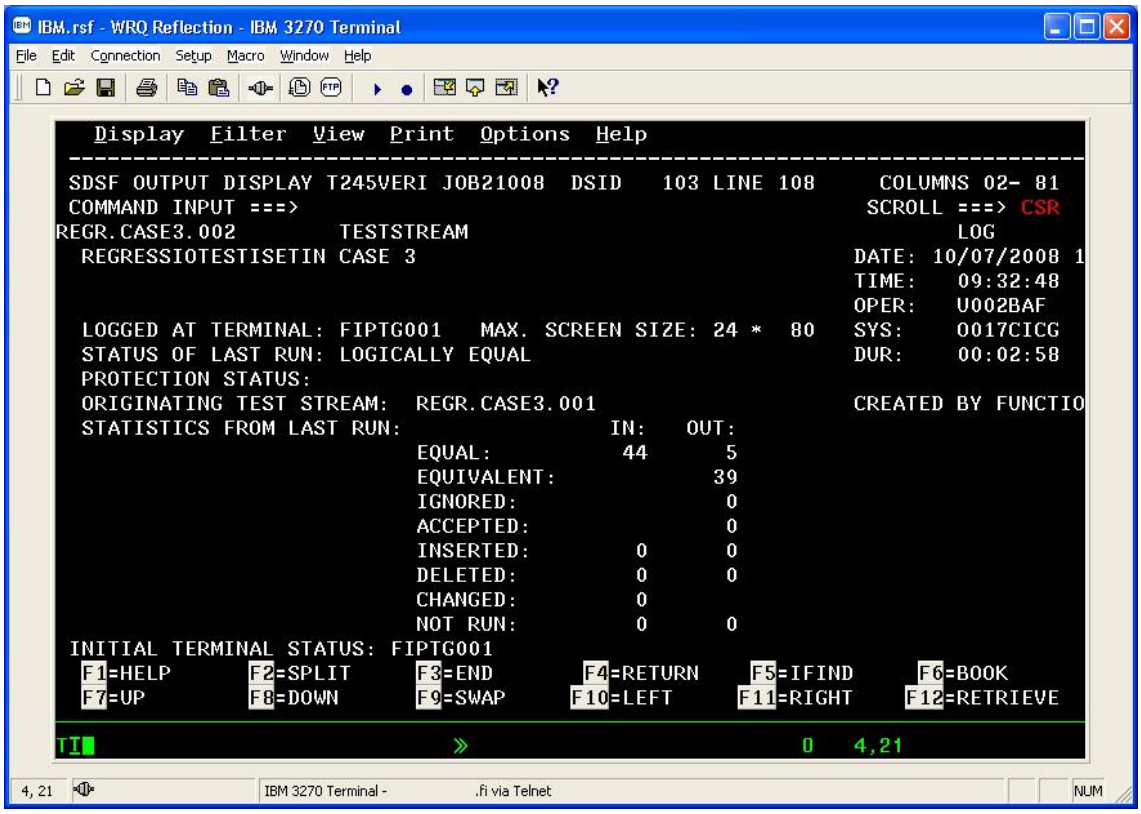
Output JCL

Jos haluat saada raportin tulostettuna, aja G-kannan JCL-kirjastosta AVERIF(T245VERI). Lisää ajoon haluamasi test streamin nimi.

Mene TSO:ssa 2.sd.h -> pre t245veri ja laita merkki ? ajon eteen. Näin päästään ajon ositetuun näkymään josta voidaan valita eri toimintoja.




Valitse TCAPRINT ja tulosta raportti lokilta. Raportti näyttää kuvan mukaiselta.




Raportin kautta voidaan tarkastella testien lopputuloksia ja yksityiskohtia. Jos testitulokset ajaa JCL- eräajolla tähän muotoon, ne voidaan myös tulostaa.